

## OBJETIVOS MÍNIMOS

1. Saber instalar nuevas aplicaciones y conocer los distintos sistemas de gestión de paquetes en Linux.
  2. Poder configurar de manera sencilla el cortafuegos.
  3. Poder acceder a un servidor de manera segura con ssh.
  4. Conocer una interfaz web para administrar servicios.
  5. Configurar para un funcionamiento básico un servidor web en Windows y Linux.
- 

## 1. Instalando software

### 1.1. Yum y apt, sistema de firmas.

- Instalación, búsqueda y desinstalación de paquetes.

Para instalar un paquete, el siguiente comando es utilizado:

```
sudo yum install $packageName | apt-get install $packageName
```

Para buscar un paquete instalable:

```
sudo yum search $packageName | apt-cache search $packageName
```

Asimismo, para quitar un paquete, el comando es:

```
sudo yum remove|erase $packageName | apt-get remove $packageName
```

YUM no incluye un comando autoremove para la búsqueda y eliminación de las dependencias no utilizadas, sin embargo, incluye una gran característica para la instalación de un paquete a partir de una url, que Apt no incluye:

```
sudo yum install $url
```

- Añadir nuevos repositorios.

Para añadir nuevos repositorios a yum es necesario añadir los archivos de definición de dichos repositorios a los archivos de yum:

```
/etc/yum.repos.d/ o /etc/yum/repos.d/
```

Los archivos de definición suelen ser proporcionados por los proveedores de paquetes. Por ejemplo, para el repositorio `elasticsearch.repo`, es necesario crear un archivo `/etc/yum/-repos.d/elasticsearch.repo` con lo siguiente:

```
[elasticsearch - 2.x ]
name=Elasticsearch repository for 2.x packages
baseurl=http://packages.elastic.co/elasticsearch/2.x/centos
gpgcheck=1
gpgkey=http://packages.elastic.co/GPG-KEY-elasticsearch
enabled=1
```

Este proceso se puede agilizar mediante el comando:

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

Se puede ver como el repositorio ya es detectado por yum mediante el siguiente comando:

```
yum repolist
```

- Sistema de firma en los paquetes.

Además de la ubicación del depósito, el archivo `repo` nos dice si un depósito en particular se encuentra habilitado y si las firmas GPG deberían utilizarse para verificar los paquetes descargados.

```
/etc/yum.repos.d/*.repo
```

```
[ian@echidna ~]$ cat /etc/yum.repos.d/fedora-updates.repo
[updates]
name=Fedora $releasever - $basearch - Updates
failovermethod=priority
#baseurl=http://download.fedoraproject.org/pub/fedora/linux/updates/$releasever/$basearch/
mirrorlist=https://mirrors.fedoraproject.org/metalink?repo=updates-released-f$releasever&arch=$basearch
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-fedora-$basearch
```

Instalación de paquetes con resolución automática de dependencias, sin verificación de firmas digitales:

```
yum --nogpgcheck localinstall packagename.arch.rpm
```

A partir de la versión 0.6, apt comenzó a usar criptografía para validar los paquetes descargados, a esto se le llama comúnmente "**Secure Apt**" (o "apt-secure").

La **criptografía de llave** (o clave) pública se basa en el par de llaves: una llave pública y una privada. Si bien la llave pública se da a todo el mundo, la llave privada debe permanecer en secreto, así es posible usar una llave privada para firmar un archivo (no para encriptarlo) y cualquiera que tenga la llave pública puede comprobar que el archivo fue firmado por esa llave, esto nos asegura que nadie que no tenga esa llave puede falsificar la firma.

**GPG** ([GNU Privacy Guard](#)) es la herramienta que se utiliza en Secure-Apt para firmar los ficheros y comprobar sus firmas. GPG cifra los mensajes usando pares de claves individuales asimétricas generadas por los usuarios. Las claves públicas pueden ser compartidas con otros usuarios de muchas maneras, un ejemplo de ello es depositándolas en los [servidores de claves](#).

**Apt-key** es un programa que se usa para gestionar el anillo de llaves de gpg para asegurar Apt. El anillo de llaves se guarda en el archivo /etc/apt/trusted.gpg (no confundir con el archivo /etc/apt/trustdb.gpg con el que está relacionado). Apt-key puede ser usado para mostrar las llaves en el anillo de llaves, y para añadir o remover una llave.

Así si un paquete viene de un archivo sin firma o con una firma de la que apt no tiene una clave, se considerará como no confiable y se nos mostrará la correspondiente advertencia, es decir, apt-get (actualmente) sólo nos advierte de los archivos sin firmar pero no impide su descarga y posterior instalación. Sin embargo para los paquetes que vienen firmados, APT utiliza este sistema de criptografía GPG para validar los .deb descargados y asegurarse de que no han sido alterados en modo alguno.

## 1.2. Convergencia hacia un gestor paquetes: DNF y Snap.

DNF como sucesor o versión de próxima generación de YUM. DNF significa YUM Dandified. DNF se introdujo por primera vez en Fedora 18 y reemplazó a YUM por completo en Fedora 22.

Snap funciona diferente, tiene varias cosas que lo hace completamente diferente a los .deb o a los .rpm, los Snap incluyen todas las dependencias en un solo paquete. Esto aporta bastantes ventajas, como por ejemplo, que se puede instalar en cualquier Ubuntu sin importar su versión (de 16.04 en adelante, claro). A esto se le añade que al no utilizar las librerías del sistema, se encuentra aislado del resto, esto hace que sean mas seguros que el resto porque **no alteran el sistema**.

## 2. Gestionando el cortafuegos: reglas para iptables.

- Apertura/cierre de puertos y comprobación de estado.
- Escaneo de puertos con nmap.

- Ubuntu Server: ufw (Uncomplicated FireWall).

```
sudo apt-get install ufw
sudo ufw default deny incoming #por defecto
sudo ufw default deny incoming #por defecto
sudo ufw default deny outgoing #lo contrario de lo anterior y siendo más restrictivos podemos luego precisar qué abrir.
sudo ufw allow ssh #por supuesto, de otro modo nos quedaríamos sin poder volver a entrar :)
sudo ufw allow www
sudo ufw allow ftp
```

Y para borrar reglas:

```
sudo ufw delete allow ftp
```

Una vez ufw configurado, lo usaremos con los siguientes comandos:

```
sudo ufw enable
sudo ufw status #veremos el estado
sudo ufw disable #pararlo
sudo ufw reset #resetearlo y comenzar la configuración a partir de la de por defecto
```

Hasta aquí hemos manejado como loguearnos y asegurar los puertos de la máquina, pero no estamos 'limitando' que uso se hace de estos puertos. Es decir, se podría intentar entrar por fuerza bruta, aunque para evitarlo usaremos fail2ban. De ese modo, es posible controlar los intentos de login en diferentes medios al servidor.

A nivel avanzado, gestiona un alto nivel de personalización, permitiendo vigilar diferentes logs de sistema operativo.

- CentOS: firewall-cmd.

Es muy importante conocer todo lo que un Firewall nos ofrece a nivel de protección y es importante saber que en CentOS 7 la solución incluida a nivel de Firewall es llamada Firewallld la cual nos ofrece las siguientes ventajas.

→ Ventajas Firewallld

Las ventajas del Firewall de CentOS 7 son:

Es un cortafuegos dinámico.

Estable.

Múltiples opciones de configuración.

Soporta configuraciones Ipv4, Ipv6 y puentes de Ethernet.

Podemos definir diversas formas de configuración de Firewallld (continua y en ejecución)

Analizaremos en detalle cómo funciona Firewallld en CentOS 7 y de esta manera comprenderemos todo su gran alcance.

Es importante que, antes de crear las reglas necesarias con Firewallld, activemos el servicio de Firewallld. Para ello ingresamos lo siguiente:

```
sudo systemctl start Firewallld.service
```

Una zona de red es aquella cuya función es **definir el nivel de confianza** que tendrá la conexión de red.

Estas zonas son **administradas por Firewallld** en diversos grupos de reglas y una zona puede ser usada por muchas conexiones de red.

Drop - Block - Public - External - DMZ - Work - Home - Internal - Trusted

### **Cómo abrir un puerto para una zona concreta en CentOS 7**

```
sudo Firewall-cmd --zone=public --add-port=3500/udp
```

Para ver los puertos abiertos en el Firewall podemos usar el siguiente comando.

```
Firewall-cmd --list-ports
```

Para ver el estado del servicio de Firewall usaremos el siguiente comando.

Podemos ver que su estado es **running (Corriendo)**. De este modo hemos habilitado

el servicio y estamos en condiciones de crear y editar las reglas del Firewall en CentOS 7.

```
Firewall-cmd -state
```

Escaneo de puertos con nmap

### **Escaneo silencioso (TCP SYN scan) sin ping**

```
# nmap -sS -PN <TARGET>
```

TCP SYN scan es tal vez la técnica de escaneo más básica y utilizada. Consiste en iniciar una sesión TCP, enviando el flag SYN, y examinar la respuesta del servidor (SYN-ACK) pero no continuar el protocolo (no enviar nunca el paquete TCP con el flag ACK). De esta forma la sesión nunca se establece, ya que la conexión nunca se completa. Esto tiene un par de beneficios: primero que el scan es rápido, pues se minimiza el número de paquetes enviados hacia el *target*; segundo que es una técnica *stealth* (por no completar la conexión).

Ejemplo:

```
root@linuxito:~# nmap -sS -PN www.mongochito.xyz
```

```
Starting Nmap 7.01 ( https://nmap.org ) at 2017-06-28 08:28 EDT
Nmap scan report for www.mongochito.xyz (192.168.129.241)
Host is up (0.019s latency).
Not shown: 997 filtered ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
443/tcp   open  https
```

```
Nmap done: 1 IP address (1 host up) scanned in 5.62 seconds
```

El escaneo lleva unos 5 segundos y se observa que el servidor *target* ("www.mongochito.xyz") tiene abiertos los puertos 22, 80 y 443.

Se evita determinar si el host está "vivo" utilizando ping, ya que muchos firewalls impiden el tráfico del protocolo ICMP.

Por defecto, la salida de Nmap muestra aquellos "puertos interesantes". En general se trata de aquellos puertos que están abiertos en el host *target*.

### 3. Administración remota: SSH (cliente y servidor).

→ Instalar, configurar y asegurar el servicio SSH, limitando el acceso por contraseña y del usuario root.

```
sudo apt install ssh
```

Secuencia de comandos y modificaciones a los archivos correspondientes para permitir acceder a la consola remota sin introducir la contraseña. (Pistas: ssh-keygen, ssh-copy-id).

Es posible prescindir de la contraseña, pero no de la seguridad. Por ello, el mejor modo para que esto se cumpla, sin necesidad de que haya que introducir constantemente la contraseña, es utilizar claves privadas y públicas (keys). Esto es más seguro que utilizar una contraseña, ya que se previenen algunos tipos de ataques, únicamente puede acceder al servidor los ordenadores que posean la llave.

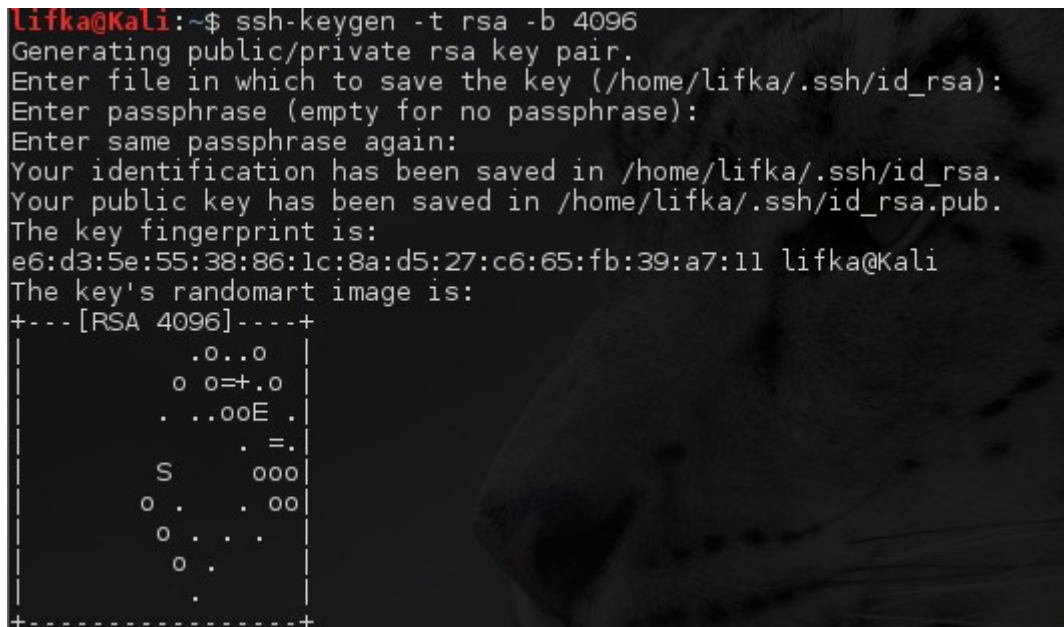
Se utiliza una clave pública, y otra privada. La privada se sitúa en la máquina que actúa como cliente, de modo que solo la posee la máquina autorizada a conectarse al servidor.

La pública se entrega al servidor que va a recibir conexiones de esa máquina.

El primer paso es generar las claves desde la máquina local (la que va a conectarse por ssh).

```
ssh-keygen -t [tipo de encript] -b [bits]
```

Por ejemplo: ssh-keygen -t rsa -b 4096



```
lifka@Kali:~$ ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/lifka/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/lifka/.ssh/id_rsa.
Your public key has been saved in /home/lifka/.ssh/id_rsa.pub.
The key fingerprint is:
e6:d3:5e:55:38:86:1c:8a:d5:27:c6:65:fb:39:a7:11 lifka@Kali
The key's randomart image is:
+---[RSA 4096]---+
|          .o..o |
|         o o+=.o |
|        . ..ooE .|
|           . =.  |
|       S        ooo|
|    o . . . oo  |
|    o . . .    |
|    o . .      |
|    .           |
+-----+-----+
```

Y, a continuación, entregarle la llave pública al servidor:

ssh-copy-id usuario@host

```
lifka@Kali:~$ ssh-copy-id -p 3022 javieriv@192.168.56.1
/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
javieriv@192.168.56.1's password:
Number of key(s) added: 1
Now try logging into the machine, with:  "ssh -p '3022' 'javieriv@192.168.56.1'
"
and check to make sure that only the key(s) you wanted were added.
```

Y la registramos, en el servidor remoto, utilizando:

```
$ssh-add keyname
```

A partir de este momento se podrá realizar la conexión sin introducir la contraseña, ya que el servidor puede reconocer al cliente.

¿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.

El archivo que contiene la configuración de sshd (daemon de ssh) es /etc/ssh/sshd\_config

Hay que modificar el parámetro PermitRootLogin

```
PermitRootLogin no
```

Para cambiar el puerto hay que configurar el parámetro:

```
Port puerto
```

Y para que esto sea efectivo, reiniciar el servicio:

```
sudo service ssh restart
```

En CentOS, es igual de sencillo:

```
service nombre_servicio restart
```



Y tras reiniciar el servicio, como se aprecia en la imagen, es posible acceder correctamente:

```
lifka@Kali:~$ ssh javieriv@192.168.56.3 -p 3049
The authenticity of host '[192.168.56.3]:3049 ([192.168.56.3]:3049)' can't be est
blished.
ECDSA key fingerprint is a8:ed:ed:51:42:6c:6b:db:46:3b:aa:67:f0:46:c9:cf.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '[192.168.56.3]:3049' (ECDSA) to the list of known hos
s.
Welcome to Ubuntu 14.04.3 LTS (GNU/Linux 3.19.0-33-generic x86_64)

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

System information as of Fri Nov 13 06:34:42 CET 2015

System load:      0.0          Processes:         132
Usage of /home:   6.4% of 451MB Users logged in:     1
Memory usage:    7%           IP address for eth0: 10.0.2.15
Swap usage:      0%           IP address for eth1: 192.168.56.3

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

5 packages can be updated.
5 updates are security updates.
```

### 3.1. Screen, terminator y tmux.

Gracias a screen se puede dejar una sesión a medias y luego recuperarla, o tener varias sesiones en una misma terminal.

Para utilizarlo es necesario instalar el paquete correspondiente e invocar a screen desde una terminal escribiendo screen . A continuación ya se pueden crear más terminales dentro de esa terminal y moverse entre ellas.

En las siguientes imágenes, se han creado dos sesiones con ctrl+a+c , y es posible alternar entre una terminal u otra pulsando ctrl+a:

```
lifka@Kali:~$ screen
lifka@Kali:~$ ls
Acquire::http::Proxy
Android
android-sdk-linux
AndroidStudioProjects
Descargas
Documentos
Escritorio
grep
Havij 1.15 - Advanced SQL Injection
Imágenes
Kismet-20151106-22-42-57-1.alert
Kismet-20151106-22-42-57-1.gpsxml
Kismet-20151106-22-42-57-1.nettxt
Kismet-20151106-22-42-57-1.netxml
lifka@Kali:~$
```

Utilizando screen, terminal b:

```
lifka@Kali:~$ ssh -p '3022' 'javieriv@192.168.56.1'
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 Mon Nov  9 13:55:42 CET 2015

System load:      0.0                Processes:        134
Usage of /home:   0.7% of 451MB      Users logged in:  1
Memory usage:     4%                IP address for eth0: 10.0.2.15
Swap usage:       0%

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

Last login: Mon Nov  9 13:55:42 2015 from 10.0.2.2
javieriv@UbuntuServer:~$
```

Pero su principal función es guardar las sesiones abiertas, aunque cerremos la terminal podemos recuperar la sesión más tarde.

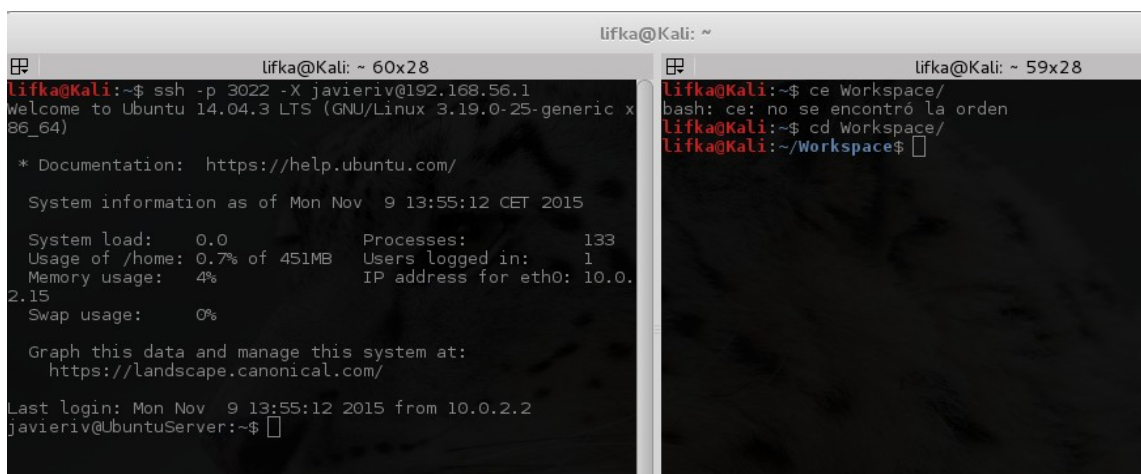
Listar sesiones activas:

```
screen -ls
```

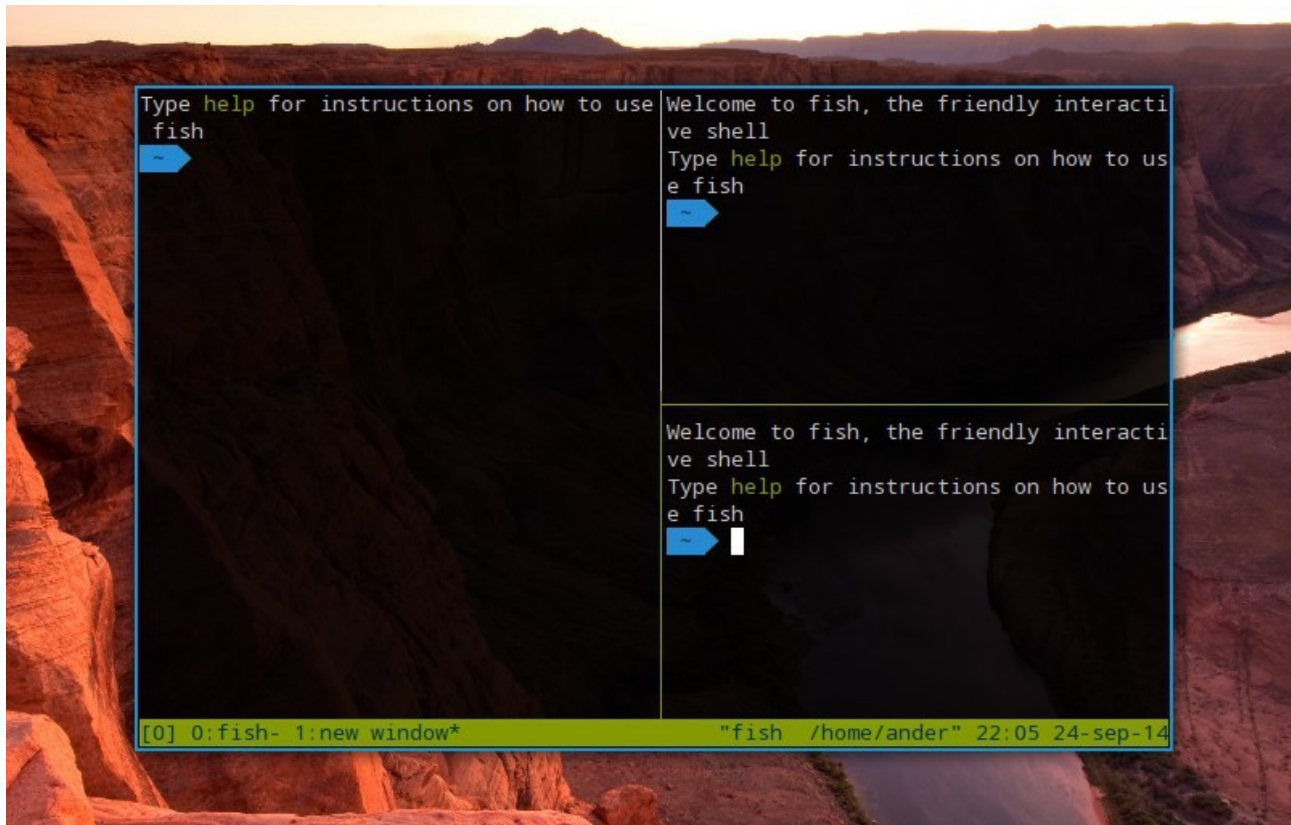
Seleccionar una sesión:

```
screen -r sesion
```

Terminator emula una terminal de pantalla en el interior de su X-Window y puede dividirse en otras consolas. Es decir, tenemos una ventana que engloba a nuestra primera consola, pero dicha ventana puede dividirse en dos consolas, y cada una de ellas en otras dos, etc... permitiendo divisiones tanto verticales como horizontales.



La renovación natural de screen es [tmux](#), que viene a ser un "multi-plexer" con funciones diversas. Entre otras: pestañas, paneles, poder administrar diferentes sesiones en una misma ventana, trabajo cooperativo vía SSH, etc.



Una vez instalado podremos iniciarlo escribiendo `tmux` en nuestro emulador de terminal favorito. Veremos que aparecerá una línea de color verde en la parte inferior de nuestra [terminal](#). En ella, aparecerá un `[0]`, que indica que estamos en la sesión 0 (la numeración empieza de 0 hacia arriba). A su derecha, veremos las pestañas (tmux los llama ventanas, pero se ven como pestañas).

Lo primero que tenemos que saber es **el comando básico a través del cual se ejecutarán las demás ordenes**. Dicho comando es la combinación de teclas `ctrl+b`. Una vez pulsada esa combinación podremos ejecutar diferentes acciones como por ejemplo, **añadir un panel** vertical pulsando `%` u horizontal pulsando `"`. Podremos **crear nuevas pestañas** (ventanas según tmux) con `c`, darles un nombre con `,`, movernos por ellas con `n` o `p` y cerrarlas con `x`.

Pero **mi característica favorita es poder ejecutar un comando y ser capaces de salirnos de la terminal para ejecutar otra tarea**. Esto es fantástico si estamos administrando un servidor, ya que solo tenemos una terminal. Para ello, qué mejor que poner un ejemplo práctico. Supongamos que estamos haciendo alguna tarea de copia de [seguridad](#) o simplemente actualizando el sistema. Si pulsamos después de `ctrl+b` la tecla `d`, dejaremos la sesión (el comando seguirá ejecutándose en segundo plano) y podremos iniciar otra sesión diferente. Podremos retomar la sesión con `tmux` y si esto lo hacemos vía SSH en el PC de

otra persona, si esa persona entra a la sesión **seréis capaces de usar colaborativamente la terminal**, al más puro estilo [Google Docs](https://docs.google.com).

3.2. Un poco de seguridad: fail2ban y rkhunter.

Gracias a Fail2ban me quité de encima algunas IP chinas que no paraban de llenarme los logs :).

```
apt-get install fail2ban
```

Su instalación nos crea una serie de ficheros:

```
/etc/fail2ban/fail2ban.conf (basic settings)
/etc/fail2ban/jail.conf      #configuración de lo que se va a monitorizar, por
defecto SSH
/etc/fail2ban/action.d/      #IP sospechosas sin bloquear, de momento.
/etc/fail2ban/filter.d/      #configuración de filtros para detectar los ataques
```

Por defecto fail2ban usa /etc/fail2ban/jail.conf , es preferible utilizar jail.local:

```
nano /etc/fail2ban/jail.local
```

La primera opción que nos interesa es bantime , con la que podemos establecer el tiempo (en segundos) que permanece baneado el usuario.

```
bantime = 3600
```

A continuación tendremos que localizar [ssh]. Una vez en las opciones de ssh, se puede modificar la opción maxretry , con la cual establecemos el número de intentos fallidos permitidos antes de banear la IP. Se dejará en 3.

```
maxretry = 3
```

Tras esto, es necesario iniciar el servicio. En mi caso he encontrado dos problemas, el primero de ellos se ha solucionado eliminando el archivo /var/run/fail2ban/fail2ban.sock .

El segundo de ellos era que es necesario tener permisos de superusuario para iniciar el daemon ( sudo /etc/init.d/fail2ban start ).

Fragmento de configuración correspondiente a SSH:

```
[ssh]
enabled = true
port = ssh
filter = sshd
logpath = /var/log/auth.log
maxretry = 2
```

Cuando el servicio se reinicia se pierden las IP baneadas:

```
/etc/init.d/fail2ban restart #o sudo service fail2ban restart
```

Para desbloquear una IP usar:

```
iptables -D fail2ban-local -s 55.222.222.22 -j DROP
```

Al instalarlo la primera vez, tener en cuenta lo que menciona la documentación sobre el consumo excesivo de python, para ello, aconseja configurarle:

```
ulimit -s 256
```

Si estamos ante una nueva instalación del servidor, vale la pena valorar la instalación de tripwire o aide. Estas aplicaciones realizan una 'observación' de la configuración inicial, y si esta cambia en un futuro, nos notificaría el cambio.

Antivirus (rkhunter)

Si deseamos escanear en busca de virus, rootkits, backdoors y sploits podemos usar rkhunter

```
rkhunter --check
rkhunter --update
rkhunter --propupd
rkhunter --check
rkhunter --checkall
```

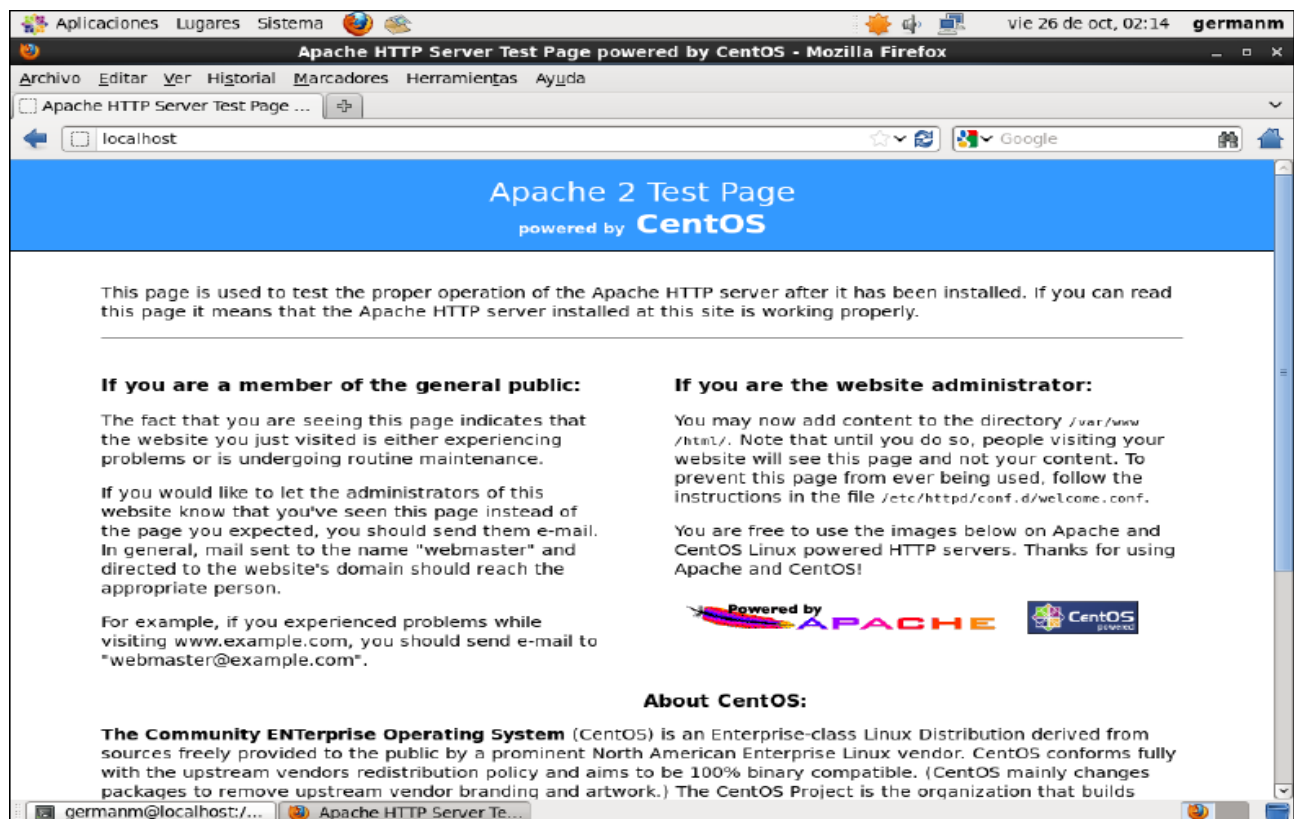
4. Instalar servidor web básico: configuración LAMPP y demostrar su funcionamiento mediante script.

**Instalación de Apache + MySQL + PHP en Linux (LAMP)**

**Muestre los comandos que ha utilizado en Ubuntu Server y en CentOS (aunque en este último puedo utilizar la GUI, en tal caso, realice capturas de pantalla)**

Para instalar Apache + MySQL + PHP seguimos los siguientes pasos:

- Primero instalamos Apache, para ello introducimos el comando correspondiente:
- Ubuntu: `# apt-get install apache2`
- CentOS: `# yum install httpd`
- Seguidamente instalamos MySQL:
- Ubuntu: `# apt-get install mysql-server`
- CentOS: `# yum install mysql-server mysql`
- Finalmente instalamos PHP:
- Ubuntu: `# apt-get install php5 libapache2-mod-php5 php5-cli php5-mysql`
- CentOS: `# yum install php php-mysql`



**Enumere otros servidores web (mínimo 3 sin considerar Apache, IIS ni nginx)**

Podemos encontrarnos otros servidores como Lighttpd, Cherokee o HTTP Explorer, teniendo todos en común que son servidores web libres.

**¿Cómo verificar el correcto funcionamiento luego de instalar un servidor LAMP?**

Primero es necesario crear una base de datos de prueba. El siguiente script crea una base de datos llamada "pepe" que contiene una única tabla con dos campos: un campo numérico "id", que funciona como clave única para la fila; y un campo "texto" que utilizo para guardar cualquier texto.

```
CREATE DATABASE IF NOT EXISTS pepe;
```

```
USE pepe;
```

```
CREATE TABLE IF NOT EXISTS prueba (  
    id BIGINT AUTO_INCREMENT,  
    texto VARCHAR(255),  
    PRIMARY KEY(id)  
) Type=InnoDB;
```

Guardar el contenido del script en un archivo db.sql:

```
root@debian6:~# cat db.sql  
CREATE DATABASE IF NOT EXISTS pepe;
```

```
USE pepe;
```

```
CREATE TABLE IF NOT EXISTS prueba (  
    id BIGINT AUTO_INCREMENT,  
    texto VARCHAR(255),  
    PRIMARY KEY(id)  
) Type=InnoDB;
```

Utilizando el cliente mysql desde línea de comandos, conectarse al servidor MySQL local con credenciales de administrador ("root"):

```
root@debian6:~# mysql -u root -p  
Enter password:  
Welcome to the MySQL monitor.  Commands end with ; or \g.  
Your MySQL connection id is 95  
Server version: 5.1.73-1 (Debian)
```

Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.



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

```
mysql>
```

Ejecutar el script db.sql para instalar la base de datos de prueba:

```
mysql> source db.sql;
Query OK, 1 row affected (0.00 sec)
```

```
Database changed
Query OK, 0 rows affected, 1 warning (0.00 sec)
```

Comprobar que la estructura de la tabla "prueba" se haya creado correctamente ejecutando describe prueba:

```
mysql> describe prueba;
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra          |
+-----+-----+-----+-----+-----+-----+
| id     | bigint(20)    | NO   | PRI | NULL    | auto_increment |
| texto  | varchar(255)  | YES  |     | NULL    |                |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

Cerrar la conexión al motor de bases de datos:

```
mysql> quit
Bye
root@debian6:~#
```

Contando con la base de datos de prueba instalada, es necesario crear un script para interactuar con la misma. El siguiente script PHP se conecta a la base de datos "pepe", inserta una nueva fila en la tabla "prueba", y finalmente muestra el contenido de la misma:

```
<?php
```

```
echo "<h4>PRUEBA</h4>";
```

```
// Parámetros de conexión a la base de datos
```

```
$host = "localhost";
```

```
$db = "pepe";
```

```
$user = "root";
```

```
$pass = "root";
```

```
$link = "";
```

```
// Conectar al servidor MySQL
```

```
if (!$link = mysql_connect($host,$user,$pass)) {
    exit("Imposible conectar a la base de datos.");
}
```

```
// Seleccionar base de datos
```



```

if (!mysql_select_db($db,$link)) {
    exit("Imposible seleccionar base de datos: ".mysql_error($link));
}

// Insertar una nueva fila en la tabla "prueba"
/*
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra          |
+-----+-----+-----+-----+-----+-----+
| id    | bigint(20)    | NO   | PRI | NULL    | auto_increment |
| texto | varchar(255)  | NO   |     | NULL    |                |
+-----+-----+-----+-----+-----+-----+
*/
$date = new DateTime("NOW");
$date = $date->format(DateTime::RFC850);
$query = "INSERT INTO prueba (texto) VALUES ('$date')";
if (!mysql_query($query,$link)) {
    exit("Error al insertar en la base de datos: ".mysql_error($link));
}

// Seleccionar datos de la tabla "pepe"
$query = "SELECT id,texto FROM prueba;";
if (!$result = mysql_query($query,$link)) {
    exit("Error al seleccionar datos de la base de datos: ".mysql_error($link));
}

// Imprimir tabla
?>
<p>Ultimos Accesos:</p>
<table style="border: 1px solid #fff;">
<tr><th>ID</th><th>Fecha</th></tr>
<?php

while ($row = mysql_fetch_array($result)) {
    echo "<tr><td>" . $row['id'] . "</td><td>" . $row['texto'] . "</td></tr>";
}

?>
</table>
<?php

// Cerrar conexión MySQL
mysql_close();

?>
<p>-- FIN --</p>

```

En Debian y derivados, el servidor Web Apache utiliza el directorio `/var/www/` como directorio de trabajo del virtual host por defecto.

Crear el directorio `/var/www/prueba/`:

```
root@debian6:~# cd /var/www/  
root@debian6:/var/www# mkdir prueba  
root@debian6:/var/www# cd prueba  
root@debian6:/var/www/prueba#
```

Crear el archivo `index.php`:

```
root@debian6:/var/www/prueba# nano index.php
```

Copiar el contenido anterior y guardar el script PHP. Notar que en el script se configuran las variables de acceso al servidor MySQL, editar según corresponda.

Asegurarse de que el servidor Apache (corre con el usuario `www-data` en Debian y derivados) tenga permisos de lectura y ejecución sobre el directorio `/var/www/prueba/` y el script `index.php`.

Finalmente es posible acceder al servidor Web desde un navegador. Por ejemplo, si la dirección IP del servidor Web es `192.168.122.245`, acceder a la URL `http://192.168.122.245/prueba/`:



En caso de errores es posible revisar los logs de accesos (`/var/log/apache2/access.log`) y de errores (`/var/log/apache2/error.log`) del servidor Web Apache.

4.1. Webmin: instalar y dejar en funcionamiento, además de citar las ventajas/desventajas de su uso vs. SSH.

**Webmin** es un panel de control que permite administrar de forma gráfica y fácil un **sistema Linux** desde el navegador web de forma remota. **Webmin** es completamente modular y desde su sitio web oficial se pueden descargar e instalar módulos que te permitirán configurar distintas partes del **sistema operativo** sin necesidad de ponerte delante de la **terminal de Linux**.

Descargamos webmin de su página oficial <http://webmin.com/> , y lo enviamos a alguno de los servidores, en este caso a Ubuntu Server:

```
scp -P 3029 Descargas /webmin_1.770_all.deb  
javieriv@192.168.56.3: /home/javieriv
```

Y, a continuación, instalamos el paquete en el servidor.

```
root@UbuntuServer:/home/javieriv# dpkg -i webmin_1.770_all.deb  
Seleccionando el paquete webmin previamente no seleccionado.  
(Leyendo la base de datos ... 79775 ficheros o directorios instalados actualmente.)  
Preparing to unpack webmin_1.770_all.deb ...  
Unpacking webmin (1.770) ...
```

Una vez instalado ya podemos acceder a este servicio a través del puerto 10000 utilizando nuestro usuario y clave.

Panel de webmin funcionando correctamente

Se ha probado a denegar el acceso a las opciones de control desde una IP concreta.

**Access control options**

Allowed IP addresses ☐ Allow from all addresses ☐ Only allow from listed addresses ☒ Deny from listed addresses

192.168.56.2

☐ Include local network in list

Resolve hostnames on every request? ☐ Yes ☒ No

Trust remote IP address provided by proxies? ☐ Yes ☒ No

IP access control using TCP-wrappers is not available, as the Authen::Libwrap Perl module is not installed. Webmin automatically install the missing Perl module.

[Return to Webmin configuration](#)

Bloqueando IP desde webmin

También se ha probado a revisar la configuración del RAID.

[Module Config](#)

## Linux RAID

Using MDADM version 3.2.5

Device name	Status	RAID level	Usable size	Member disk devices
/dev/md0	clean, degraded	RAID1 (Mirrored)	7.99 GB	/dev/sda1

**Webmin** te permitirá administrar la mayoría de las configuraciones y funciones de tu **servidor** o **VPS** sin necesidad de realizar las tareas a través de **SSH** utilizando un **cliente SSH** como **Putty**. Con él se pueden configurar aspectos internos de muchos sistemas operativos, como usuarios, cuotas de espacio, servicios, archivos de configuración, apagado del equipo, etcétera, así como modificar y controlar muchas aplicaciones libres, como el servidor web Apache, PHP, MySQL, DNS, Samba, DHCP, entre otros.

Ventajas Webmin vs. SSH

Demuestra que la Administración de Linux es fácil, con un programa que lo hace todo de manera gráfica → Webmin.

Desventajas SSH vs. Webmin

Es de notar que pese a que a veces puede sonar más *geek* el hacer todo por línea de comandos (la bendita consola que tanto miedo provoca...), uno puede confundirse y dejar al sistema colgado de hilos por un error de tipeo con lo que, necesariamente un programa que facilite la administración, siempre es bienvenido.

## 5. Copias de seguridad y control de versiones

→ Uso de tar y rsync para backups.

Existen **tres tipos distintos de copias de seguridad**:

**Completa**: Realiza una copia de seguridad de todos los archivos.

**Incremental**: Realiza una copia de los datos que se han modificado desde la última copia de cualquier tipo.

**Diferencial**: Realiza una copia de los datos que se han modificado desde la última copia completa.

**Cobian Backup** es una herramienta que permite realizar copias de seguridad en un ordenador. Es una herramienta bastante simple, no consume casi recursos y es compatible con Windows.

**Rsync** es una aplicación libre para sistemas de tipo Unix y Microsoft Windows que ofrece transmisión eficiente de datos incrementales, que opera también con datos comprimidos y cifrados.

Una característica importante de rsync no encontrada en la mayoría de programas o protocolos es que la copia toma lugar con sólo una transmisión en cada dirección. Rsync puede copiar o mostrar directorios contenidos y copia de archivos, opcionalmente usando compresión y recursión.

La función **Tar** nos permite empaquetar o desempaquetar varios archivos en uno mismo, pero no los comprime.

**tar**: comando.

**vcf**: opciones.

**v**: (verbose) muestra en pantalla las operaciones que va realizando archivo por archivo (opcional).

**c**: (create/crear) crea un archivo **tar**.

**f**: (file/archivo) indica que se dará un nombre al archivo **tar**.

**nombre\_archivo.tar**: nombre que se dará al archivo **tar**.

**nombre\_carpeta\_a\_empaquetar**: nombre de la carpeta (o del directorio) que se va a empaquetar.

o

**tar**: comando.

**vx**: opciones.

**v**: (verbose) permite obtener una descripción de los archivos. desempaquetados (opcional).

**x**: (extract/extraer) extrae los archivos.

**f**: (file/archivo) para indicar el archivo tar que contiene los archivos, parámetro siguiente.

**nombre\_archivo.tar:** el nombre del archivo **tar** de donde se extraerán los archivos.

(Para desempaquetar)

El comando **crontab** se utiliza en sistemas **UNIX** para programar la ejecución de otros comandos, es decir, para automatizar tareas. Podemos ver los crontabs que se están programados y también editarlos, lógicamente.

Para verlos, utilizamos este comando:

```
sudo crontab -l
```

Para editarlos:

```
sudo crontab -e
```

### FORMATO DE LAS TAREAS

Las tareas *cron* siguen una determinada sintaxis. Tienen 5 asteriscos seguidos del comando a ejecutar. Ahora explicaré para qué sirve cada cosa.

```
* * * * * /bin/ejecutar/script.sh
```

### Los 5 asteriscos

De izquierda a derecha, los asteriscos representan:

1. Minutos: de 0 a 59.
2. Horas: de 0 a 23.
3. Día del mes: de 1 a 31.
4. Mes: de 1 a 12.
5. Día de la semana: de 0 a 6, siendo 0 el domingo.

→ Git: control de versiones y restauración de errores.

GIT es un software de código abierto, creado en 2005 por Linus Torvalds, que nos permite crear aplicaciones permitiendo tener las versiones anteriores de estas con el fin de dar un mantenimiento más eficaz y la oportunidad tanto de corregir errores como de hacer mejoras.

### Características

- Comparando a otros sistemas VCS (subversión y compañía incluidos), su almacenamiento de información no es de forma lineal, es más bien como en forma de ficheros, con el fin de mantener la imagen de todos los archivos antes del momento de cambio.
- La mayoría de cada una de las operaciones son locales. Esto significa que

comúnmente solo se utilizarían los archivos locales, es decir del mismo servidor, no es necesario tomar información de otro equipo dentro de la red. (OJO: La mayoría no implica que siempre sea así).

- La perfecta integridad de GIT. No es posible realizar algún cambio al contenido de cualquier archivo sin que lo sepas
- No puedes recibir archivos dañados, o perder información en tránsito sin que GIT lo detecte.
- Es complicado hacer que GIT pueda deshacer y/o eliminar información, generalmente solo introduce información a las bases de datos.

Si en determinado punto el proyecto llegara a fallar, [Git](#) cuenta con la posibilidad de restaurar a un punto anterior a la realización de algún cambio, función que favorece mucho para el desarrollo e implementación del sitio ya que si por algún cambio dejó de funcionar, se puede restaurar al punto en que estaba funcionando anteriormente. una descripción más exacta sería, cada vez que realizamos algún cambio con [Git](#) tomamos una instantánea del o los archivos modificados, de esta manera cuando queramos deshacer algún cambio a nuestro proyecto, lo único que debemos hacer es restaurarlo al punto en que [Git](#) tomó la instantánea del o los archivos y comenzar de nuevo.

#### Ejemplo de restauración de errores

Todos cometemos errores, pero herramientas como Git nos viene a solucionar algunos de esos errores y problemas en lo referente al código.

Imagina que modificas un archivo y parece que todo está bien, lo añades (**git add**) y realizas un "commit" de la modificación (**git commit**).

Después de eso decides que ese archivo necesita un "último ajuste" y finalmente el archivo queda completamente irreconocible y quizás no funcionando como esperabas. **¿Qué podemos hacer?**

**Podemos recuperar el archivo** (prueba.txt en los ejemplos siguientes) **al estado en el que estaba antes de la modificación**, es decir al estado en el que estaba en el último "commit" para eso utiliza el comando **git checkout** al último commit conocido que en este caso está en HEAD:

```
git checkout HEAD prueba.txt
```

Pero si esa versión no es buena, o si quieres volver el archivo en cuestión no a esa versión si no a una anterior en el tiempo, primero deberemos **comprobar en el registro log de ese directorio de git los "commits" realizados** para poder escoger a la versión que deseamos. Echamos un vistazo a los logs con este comando:

```
git log --oneline
```

Lo que nos dará una salida "algo" similar a esta (con las lógicas diferencias de "commits", etc...)

```
79a4e5f commit prueba
f449007 segundo commit
55df4c2 Primer commit del proyecto.
```

Imaginemos que queremos devolver el archivo prueba.txt a la versión como estaba en nuestro **primer commit del proyecto**. Para ello escribiremos el siguiente comando

```
git checkout 55df4c2 prueba.txt
```

Ahora la versión antigua del archivo ha sido restaurada en el directorio de trabajo. Puedes comprobar el estado del directorio de trabajo mediante el comando `git status`. **No olvides que una vez restaurado el archivo, es necesario volver a añadir el archivo y volver a hacer un "commit"** ya que el archivó cambió.

```
git add prueba.txt
git commit -m 'restaurar prueba.txt al estado del primer commit.'
```

Comprobemos en el log de Git que efectivamente todo ha ido como queríamos:

```
git log --oneline
d512580 restaurar prueba.txt al estado del primer commit.
79a4e5f commit prueba
f449007 segundo commit
55df4c2 Primer commit del proyecto.
```

También podemos llevar no sólo un archivo a un punto predeterminado, si no **todos los archivos del repositorio**, para ello escribimos:

```
git checkout 55df4c2
```

**Al hacer esto, tu repositorio va atrás en el tiempo por completo, así que si ahora empiezas a trabajar sobre él podrías destruir tu futuro trabajo.** Por defecto Git asume que no es eso lo que quieres hacer, así que separa donde se desarrolla el trabajo HEAD del proyecto y te deja empezar a trabajar.

Aquí habría que empezar a hablar de las **ramas o "branch"** y de cómo trabajar con **ramas paralelas del proyecto**, pasar de HEAD a las ramas y finalmente unir la que deseemos.