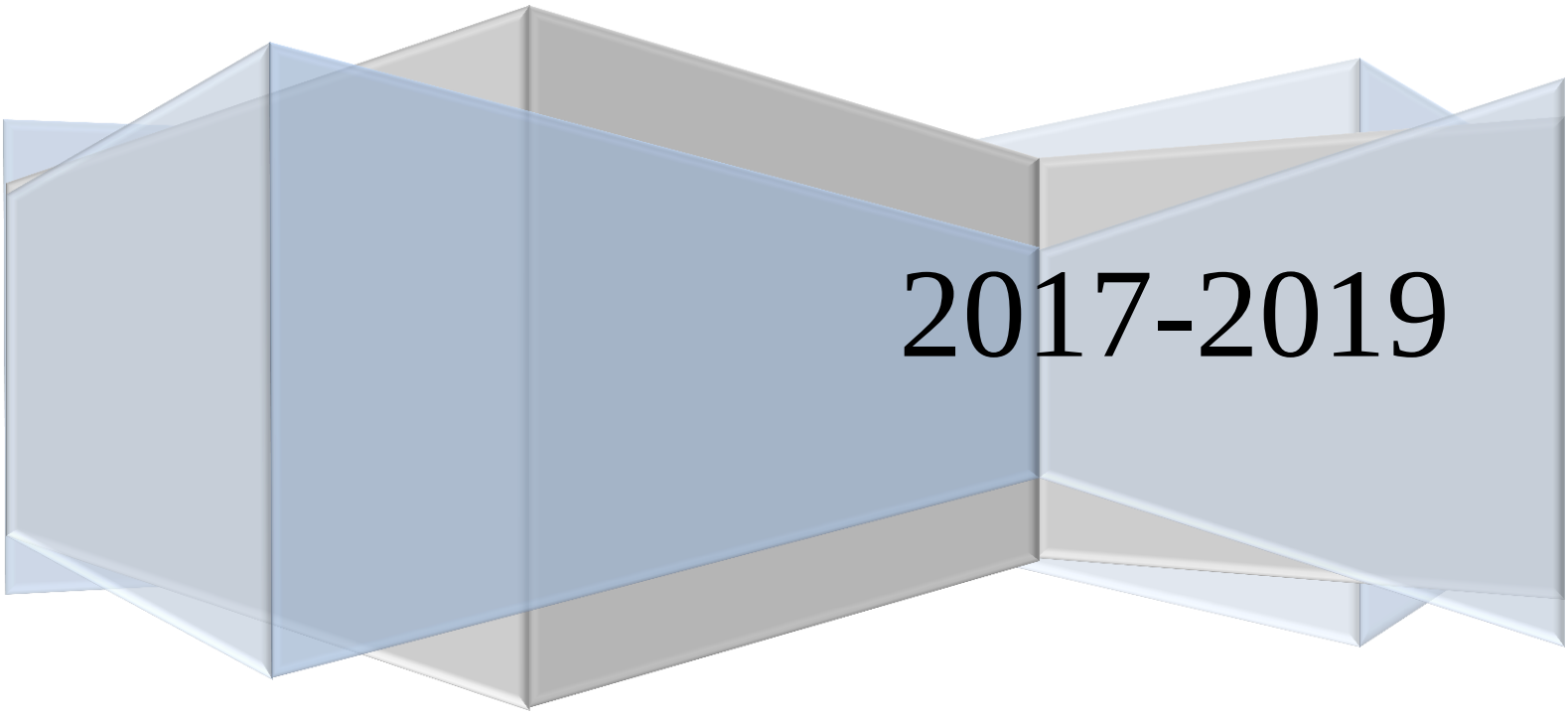
IES Universidad Laboral (Albacete)

Trabajo Fin de Grado

Mantenimiento Electrónico

Paula Romero Serrano



2017-2019

ÍNDICE

Explicación de los programas 1
- VirtualBox 1
- PuTTY 3
- PgAdmin III 4
- PgAdmin 4 5

Cómo conectarse 6
Comandos básicos PostGreSQL 9
pg_dump 11
Tipos de datos de PostgreSQL 13
pg_hba.conf y postgresql.conf 15
pg_hba.conf 15
postgresql.conf 17
Comandos básicos Script 18
chmod 22
Visualización de permisos 22
Derechos de usuarios y permisos 22
Definir los permisos de manera explícita 23
Definir los permisos utilizando referencias binarias 23
Permisos especiales 24
Heredar permisos 25
Comandos de archivos comprimidos 26
ZIP 26
TAR 27
TAR.GZ 28
7z 29
Comandos de administración de redes 30
Cron 31
Drive 33
Comandos de drive útiles 33
Scripts para hacer copia de seguridad y subirla a drive 34
Script para hacer una copia de seguridad 34
Paso a paso 35
Script para subir la copia de seguridad al drive 37
Paso a paso. 38
Configuración inicial de un servidor Linux 39

Paso 1: Sesión de root 39
Paso 2: Crear una cuenta de usuario alternativa con un reducido margen de influencia para el trabajo del día a día 39
Paso 3: Privilegios de root 40
Paso 4: Registro de prueba 40
Paso 5: Configurar un firewall básico 41
LAMP 42
Paso 1: Instalar Apache 42
Establecer `Global ServerName` para hacer que cesen las advertencias de sintaxis 42
Cortafuegos 43
Paso 2: Instalar MySQL 45
Paso 3: Instalar PHP 45
Paso 4: Probar el proceso de PHP en nuestro servidor web 47
Wordpress 49
Paso 1: Crear una base de datos y usuario para Wordpress 49
Paso 2: Instalar extensiones adicionales para PHP 50
Paso 3: Ajustar la configuración de Apache para permitir sobre-escritura y re-escritura para .htaccess 51
Habilitar sobre-escritura por .htaccess 51
Paso 4: Descargar WordPress 52
Paso 5: Configurar el directorio WordPress 53
Ajustando permisos y autoridad 53
Configurando el archivo de configuración de WordPress 54
Paso 6: Completar la instalación a través de la interfaz web 56
Actualizando WordPress 57
PHP y HTML 58
Primer código 59
Paso a paso 60
Segundo código 62
Paso a paso 63

Antes de entrar en detalles hay que hablar de los programas que vamos a utilizar. En este caso, serán:

- VirtualBox 6.0.4 (**https://www.virtualbox.org/wiki/Downloads**)
- PuTTY 0.71 (**https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html**)
- PgAdmin 4 (**https://www.pgadmin.org/download/**)
- PgAdmin III (**https://www.postgresql.org/ftp/pgadmin/pgadmin3/v1.22.2/win32/**)
- Ubuntu server 18.04.2 LTS: no es un programa pero lo incluyo en esta lista de descargas. Esto es la imagen ISO que usaremos en VirtualBox. Como dato, este sistema operativo no es GUI, es CLI, es decir, no tiene interfaz gráfica, deberemos hacer todo mediante comandos. (**https://www.ubuntu.com/download/server**)

> **Nota:** usaremos la versión 3 o 4 del PgAdmin según nos resulte más útil. Cuando usemos la versión 10 de PostgreSQL será obligatoria la versión 4 de PgAdmin. PgAdmin III se abre en una ventana aparte y PgAdmin 4 se abre en una pestaña de nuestro navegador predeterminado.

Explicación de los programas

- **VirtualBox**: es un software de virtualización para arquitecturas x86/x64, creado originalmente por la empresa alemana innotek GmbH. Actualmente es desarrollado por Oracle Corporation.
  - Con esta aplicación es posible instalar sistemas operativos adicionales, conocidos como "Sistemas Invitados", dentro de otro sistema operativo "Anfitrión", cada uno con sus recursos virtuales.
  - Entre los sistemas operativos que **soportan el modo anfitrión** se encuentran:
    - GNU/Linux.
    - Mac OS X.
    - OS/2 Warp.
    - Microsoft Windows.
    - Solaris/OpenSolaris

- Dentro de estos es posible **virtualizar** (entre otros) los sistemas operativos:
  - FreeBSD.
  - GNU/Linux.
  - OpenBSD.
  - OS/2 Warp.
  - Windows.
  - Solaris.
  - MS-DOS.
- La aplicación fue inicialmente ofrecida bajo una licencia de software privativo, pero en enero de 2007 surgió VirtualBox OSE (Open Source Edition) bajo la licencia GPL 2.
- Tiene la capacidad de emular el hardware, los discos duros de los sistemas invitados son almacenados en los sistemas anfitriones bajo una extensión propia llamada Virtual Disk Image o VDI, la cual es incompatible con otros softwares de virtualización. Otra funciones que presenta es montar imágenes ISO como unidades virtuales ópticas de CD o DVD.
- Actualmente, VirtualBox se encuentra en su versión más reciente, la 4.3.22 que incluye:
  - Soporte para X.Org Server 1.17
  - Se evita el ocultamiento no deseado de pantallas de huéspedes en invitados con varios monitores
  - Se añade compatibilidad para Linux 2.4 de 64 bits
  - Soporte para Linux 3.19
  - Soporte para Windows 10 preview
  - Se solventan los errores que se originaban al tener SMAP activado (solo para Broadwell y superior)
  - Se solventa un error en DirectSound cuando el host no encuentra ningún dispositivo de entrada

- **PuTTY**: es un cliente SSH y Telnet con el que podemos conectarnos a servidores remotos iniciando una sesión en ellos que nos permite ejecutar comandos. El ejemplo más claro es cuando empleamos PuTTY para ejecutar comandos en un servidor VPS y así poder instalar algún programa o configurar alguna parte del servidor.
  - Con PuTTY conseguimos abrir una sesión de línea de comandos en el servidor remoto para administrarlo.
  - PuTTY se puede descargar directamente desde su página oficial, que nos permite descargar PuTTY gratis e incluso otras aplicaciones complementarias.
  - **Ventajas de PuTTY:**
    - Es gratuito y de código abierto.
    - Disponible para Windows y Linux.
    - Es una aplicación portable.
    - Interfaz sencilla y manejable.
    - Muy completo y ofrece una gran flexibilidad con multitud de opciones.
    - Está en constante desarrollo.

- **PgAdmin III**: es una aplicación gráfica para gestionar el gestor de bases de datos PostgreSQL, siendo la más completa y popular con licencia Open Source.
  - Está escrita en C++ usando la librería gráfica multiplataforma wxWidgets, lo que permite que se pueda usan en Linux, FreeBSD, Solaris, Mac OS X y Windows.
  - Es capaz de gestionar versiones a partir de la PostgreSQL 7.3 ejecutándose en cualquier plataforma, así como versiones comerciales de PostgreSQL como Pervasive Postgres, EnterpriseDB, Mammoth Replicator y SRA PowerGres.
  - Está diseñado para responder a las necesidades de todos los usuarios, desde escribir consultas SQL simples hasta desarrollar bases de datos complejas.
  - El interfaz gráfico soporta todas las características de PostgreSQL y facilita enormemente la administración.
  - La aplicación también incluye un editor SQL con resaltado de sintaxis, un editor de código de la parte del servidor, un agente para lanzar scripts programados, soporte para el motor de replicación Slony-I y mucho más.
  - La conexión al servidor puede hacerse mediante conexión TCP/IP o Unix Domain Sockets (en plataformas *nix), y puede encriptarse mediante SSL para mayor seguridad.

- **PgAdmin 4**: esta versión es la que vamos a utilizar con la versión 10 de PostgreSQL ya que no tenemos compatibilidad con PgAdmin III.
  - Implementa las siguientes tecnologías:
    - Flask para el Backend,
    - Soporte Python en sus versiones: 2.7.x y 3.0 a 3.4,
    - Javascript/Jquery/Backbone para el FrondEnd,
    - Bootstrap
  - A diferencia a su versión anterior que está desarrollada con C++, esta está desarrollada con Python y sirve para gestionar el gestor de BD de PostgreSQL, y es considerada una de las más completas y populares con licencia Open Source. Además, está disponible en diferentes idiomas.
  - PgAdmin 4 nos permite acceder a todas las funcionalidades de la base de datos, consulta, manipulación y gestión de datos, incluso opciones avanzadas como manipulación del motor de replicación Slony-I. Esta puede ser ejecutas en múltiples plataformas como:
    - Linux
    - FreeBSD
    - Solaris
    - Mac OS X
    - Windows
  - Su interfaz gráfico soporta todas las características de PostgreSQL y facilita de gran manera la administración, ya que nos permite desde hacer búsquedas SQL hasta desarrollar toda nuestra base de datos de forma muy fácil e intuitiva: directamente desde la interfaz gráfica.

  - **Mejoras de PgAdmin4 respecto a PgAdmin III:**
    - Uno de los cambios más visibles, es el gran cambio visual que incluye un conjunto de iconos actualizados y fuentes incrustadas para mostrar una apariencia coherente en todas las plataformas.
    - El Query Tool y la Edit Grid fueron fusionadas en una sola herramienta.
    - La interfaz de usuario es mucho más flexible, permite pestañas para contener y reorganizar en más aspecto que anteriormente.
    - Una interfaz de usuario mucho más atractiva, haciendo uso de controles agrupados y de regiones expansibles para hacer las cosas más fácil de entender.
    - Se ha rediseñado algunos de los paradigmas de interfaz de usuario.
    - Quedó atrás la lista de controles con botones de añadir y quitar, las cuales fueron reemplazados con lo que llamamos grilla de sub-nodos que permite la edición de las clases y sus valores, con el mayor detalle disponible cuando se necesita mediante filas expandibles.

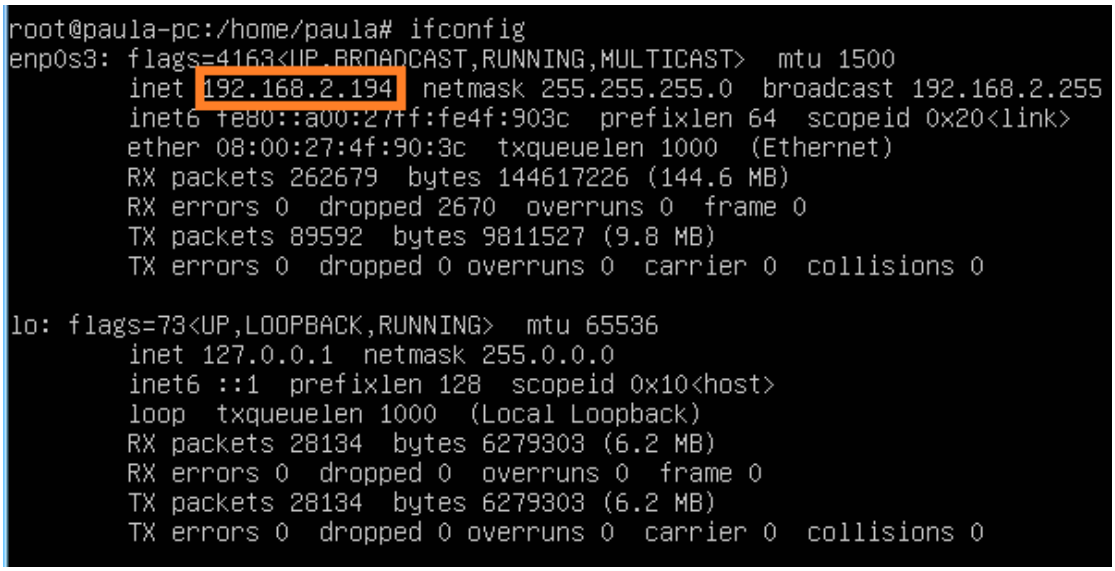
Cómo conectarse

Teniendo esto descargado, voy a explicar para qué se usará cada programa. En VirtualBox vamos a usar la imagen ISO descargada para utilizar esta distro de Linux con PostgreSQL. Por lo tanto, lo primero tras tener en marcha VirtualBox será instalarlo.

Esto lo haremos con:

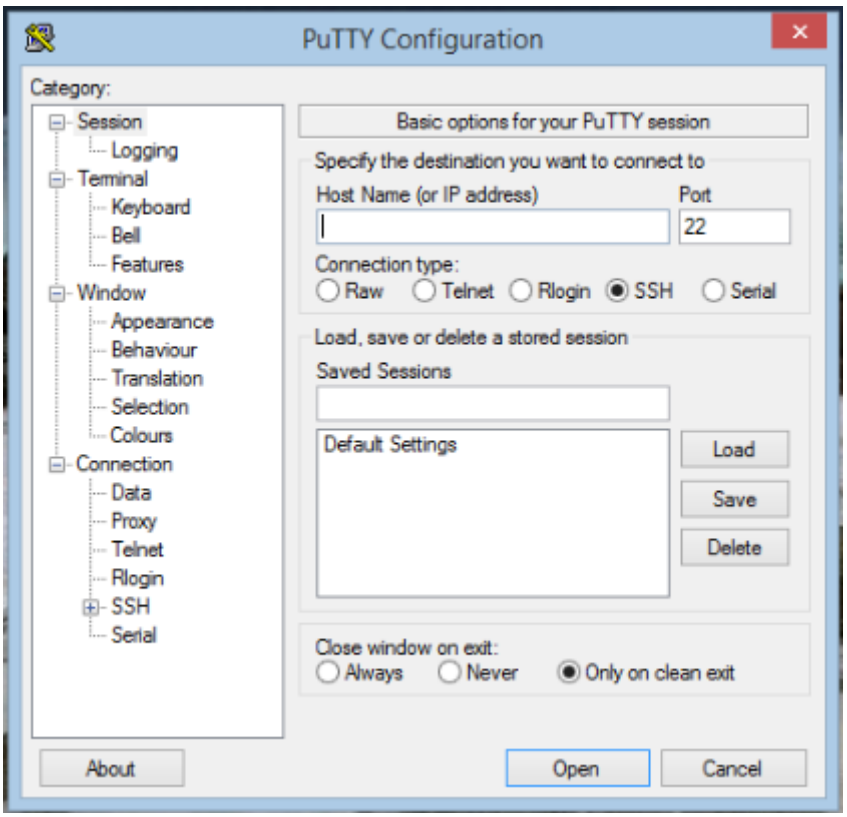
$ sudo apt-get install postgresql postgresql-contrib.

Y con "ifconfig" miraremos la IP que tenemos en el servidor. En este caso observamos que es 192.168.2.194.

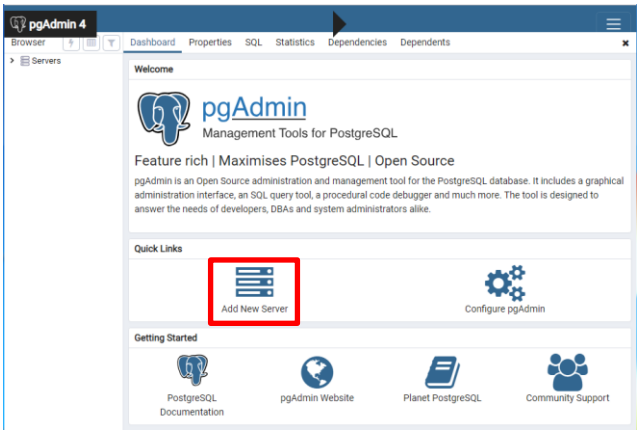


Ahora vamos a utilizar el programa PuTTY para conectarnos mediante SSH a este servidor para trabajar directamente desde este terminal en vez de con VirtualBox directamente.

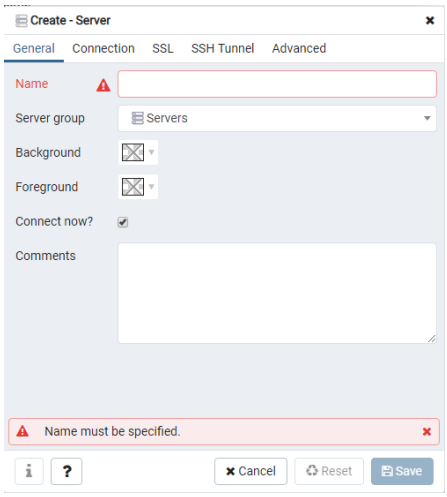
En PuTTY encontramos esta ventana al empezar. Debemos rellenar con la IP del servidor (192.168.2.194) donde pone “Host name (or IP address)” y seleccionar SSH en “Connection type”. Tras esto, ya estaremos conectados en cuanto seleccionemos “Open”.

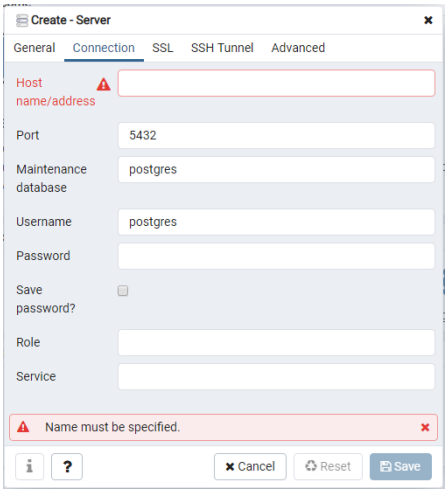


Lo siguiente será sincronizar el programa PgAdmin 4 (que al ejecutarlo se abrirá directamente una pestaña en tu navegador predeterminado) con la base de datos de PostgreSQL.



Le deberemos dar a “Add New Server”.





En el campo "Name" le tenemos que poner el nombre que queramos al servidor, es para que nosotros lo reconozcamos.
En "Host name/address" la dirección IP de nuestro servidor. Recordemos que en nuestro caso era 192.168.2.194.
En "Maintenance database": postgres.
En "Username": postgres.
En "Password": postgres.

Para que nos permita conectarnos, debemos ir primero al archivo "`pg_hba.conf`" que se encuentra en el directorio `/etc/postgresql/10/main/pg_hba.conf` del servidor. Aquí, debemos editar el archivo gracias al editor "`Nano`" en este caso, ya que es un editor más sencillo y que cuenta con los comandos necesarios para lo que queramos en la parte de abajo, al contrario de "`Vim`", que te los tienes que saber. Ejecutamos el comando "`nano pg_hba.conf`" en el directorio y lo dejamos tal y como muestra la siguiente imagen:

```
local           all             postgres                peer

# TYPE  DATABASE        USER            ADDRESS                 METHOD
local           replication     postgres                peer

# "local" is for Unix domain socket connections only
#local   all             all                                     peer
local           all             all                     md5

# IPv4 local connections:
host            all             all             127.0.0.1/32            md5

# IPv6 local connections:
host            all             all             ::1/128                 md5
# Allow replication connections from localhost, by a user with the
# replication privileges
host            all             all             192.168.2.0/24          md5
```

Nota: la línea "`host all all 192.168.2.0/24`" deberá cambiarse según la IP de nuestro servidor.

Posteriormente se hablará tanto del archivo "`pg_hba.conf`" como de este comando.

Comandos básicos PostGreSQL

Aquí voy a escribir algunos comandos útiles a la hora de administrar nuestra base de datos:

El comando para conectarse es:

```
# psql -h ip_servidor -U usuario -d base_de_datos
```

En nuestro caso:

```
# psql -h 192.168.2.194 -U paula -d mi_db
```

Iniciar sesión usuario postgres:

```
$ sudo su - postgres
```

Creación de un usuario:

```
CREATE USER nombre_usuario WITH password '123456'
```

Eliminar usuario:

```
DROP USER nombre_usuario
```

Crear base de datos:

```
CREATE DATABASE nombre_base_de_datos WITH OWNER nombre_usuario;
```

Eliminar base de datos:

```
DROP DATABASE nombre_base_de_datos
```

Acceder a la base de datos con un usuario determinado:

```
psql -U nombre_usuario nombre_base_de_datos
```

Conectarse o seleccionar una tabla específica:

```
\c nombre_tabla;
```

Obtener ayuda:

```
\h
```

Salir:

```
\q
```

Crear tabla

```
create table NOMBRETABLA(
    NOMBRECAMPO1 TIPODEDATO,
    ...
    NOMBRECAMPON TIPODEDATO
 );
```

Para ver la estructura de una tabla consultaremos una tabla propia del PostgreSQL:

```
SELECT table_name,column_name,udt_name
    FROM information_schema.columns WHERE table_name = 'nombre_tabla';
```

A la salida aparece el nombre de la tabla, los nombres de columna y el tipo de contenido de cada columna:

```
table_name      column_name     udt_name
mitabla         nombre          varchar
```

Cambiar el dueño (owner) de una tabla:

```
ALTER TABLE <nombre_tabla> OWNER TO <usuario>
```

Insertar (insert) en PostgreSQL

```
INSERT INTO nombre_tabla (columna1, columna2, columna3) VALUES (valor1,
valor2, valor3);
```

Insertar más de un registro a la vez:

```
INSERT INTO nombre_tabla (columna1, columna2, columna3) VALUES (valor1,
valor2, valor3), (otroValor1, otroValor2, otroValor3), (otroValorMas1,
otroValorMas2, otroValorMas3);
```

Lista de bases de datos:

```
\l
```

Lista de tablas en una base de datos:

```
\d
```

Conectarse o seleccionar una base de datos específica:

```
\c nombre_base_de_datos
```

Mostrar usuarios:

```
\du
```

Resetear una contraseña de usuario como administrador:

```
ALTER USER usuario_del_cambio WITH password 'contrasena_nueva';
```

Cambiar el dueño de una base de datos:

```
ALTER DATABASE nombre_base_de_datos OWNER TO nuevo_dueno;
```

pg_dump

Si hablamos de bases de datos, quizás uno de los comandos más importantes es `pg_dump`:

`pg_dump`
Esta orden de PostgreSQL extrae una base de datos en un archivo script u otro tipo de archivo.

Descripción:
`pg_dump` es una herramienta para hacer copias de seguridad a una base de datos en PostgreSQL. Hace copias de seguridad sólidas incluso si la base de datos se está usando de forma concurrida. `pg_dump` no bloquea a otros usuarios en cuanto al acceso a la base de datos (lectores o escritores).

Los volcados de memoria pueden extraerse en scripts o en forma de otro tipo de archivo. Los volcados de memoria en forma de scripts son archivos de texto planos que contienen los comandos SQL necesarios para reconstruir la base de datos al estado en el que estaba en el momento de guardar la copia de seguridad. Para restaurar desde el script, se usa psql. Los scripts pueden ser usados para reconstruir la base de datos también en otros dispositivos y en otras arquitecturas; con algunas modificaciones, incluso en otros productos de base de datos SQL.

Las alternativas de formatos de archivo tendrán que ser usados con `pg_restore` para reconstruir la base de datos. Permiten que `pg_restore` sea selectivo acerca de lo que se restaura, o incluso para reordenar los elementos antes de ser restaurados. Los formatos de archivo están diseñados para ser portables en las distintas arquitecturas.

Cuando se usa con uno de los formatos de archivo y es combinado con `pg_restore`, `pg_dump` proporciona un archivado flexible y un mecanismo de transferencia. `pg_dump` puede ser usado para hacer copias de seguridad de una base de datos, luego `pg_restore` puede ser usado para examinar el archivo y/o seleccionar qué partes de la base de datos se restaurarán.

Cuando se ejecuta `pg_dump`, se deberá examinar la salida para cualquier advertencia (se mostrará por pantalla como un error estándar).

Dump db a un archivo:

```
$ pg_dump -U nombre_usuario nombre_base_de_datos > base_de_datos.out
```

Dump todas las bases de datos:

```
$ sudo su - postgres
$ pg_dumpall > /var/lib/pgsql/backups/dumpall.sql
```

Restaurar base de datos:

```
$ sudo su - postgres
$ psql -f /var/lib/pgsql/backups/dumpall.sql mi_base_de_datos
```

También:

```
$ psql -U postgres nombre_base_de_datos < archivo_restauracion.sql
```

La forma más sencilla de hacer un backup de una base de datos completa:

```
# pg_dump basededatos > fichero.sql
```

En el caso de tener que especificar parámetros:

`-h` para el host
`-p` para el puerto
`-U` para el usuario
`-d` para la base de datos

```
# pg_dump -h host -U usuario -f directorio/nombre_archivo base_de_datos
```

La restauración del backup es el mismo proceso pero a la inversa:

```
# psql basededatos < fichero.sql
```

Backup de todas las bases de datos del servidor PostgreSQL: `pg_dumpall`

Con un único comando podemos hacer una copia de seguridad de todas las bases de datos del servidor:

```
# pg_dumpall > backup_server.sql
```

Posteriormente para restaurar todas las bases de datos:

```
# psql -f nombre_del_archivo.sql postgres
```

Tipos de datos de PostgreSQL

Tipo	Descripción
SET	conjunto de tuplas
abstime	fecha y hora absoluta de rango limitado (Unix system time)
aclitem	lista de control de acceso
bool	booleano '*true*'/'*false*'
box	rectángulo geométrico '(izquierda abajo, derecha arriba)'
bpchar	caracteres rellenos con espacios, longitud especificada al momento de creación
bytea	arreglo de bytes de longitud variable
char	un sólo carácter
cid	*command identifier type*, identificador de secuencia en transacciones
cidr	dirección de red
circle	círculo geométrico '(centro, radio)'
date	fecha ANSI SQL 'aaaa-mm-dd'
datetime	fecha y hora 'aaaa-mm-dd hh:mm:ss'
filename	nombre de archivo usado en tablas del sistema
float4	número real de precisión simple de 4 bytes
float8	número real de precisión doble de 8 bytes
inet	dirección de red
int2	número entero de dos bytes, de -32k a 32k
int28	8 números enteros de 2 bytes, usado internamente
int4	número entero de 4 bytes, -2B to 2B
int8	número entero de 8 bytes, 90#9018 dígitos
line	línea geométrica '(pt1, pt2)'
lseg	segmento de línea geométrica '(pt1, pt2)'
macaddr	Dirección MAC
money	unidad monetaria '$d,ddd.cc'
name	tipo de 31 caracteres para guardar identificadores del sistema
numeric	número de precisión múltiple
oid	tipo de identificación de objetos
oid8	arreglo de 8 *oid*s, utilizado en tablas del sistema
path	trayectoria geométrica '(pt1, ...)'
point	punto geométrico '(x, y)'

`polygon`	polígono geométrico '(pt1, ...)'
`regproc`	procedimiento registrado
`reltime`	intervalo de tiempo de rango limitado y relativo (Unix delta time)
`smgr`	manejador de almacenamiento (*storage manager*)
`text`	cadena de caracteres nativa de longitud variable
`tid`	tipo de identificador de tupla, localización física de tupla
`time`	hora ANSI SQL 'hh:mm:ss'
`timespan`	intervalo de tiempo '@ <number> <units>'
`timestamp`	fecha y hora en formato ISO de rango limitado
`tinterval`	intervalo de tiempo '(abstime, abstime)'
`unknown`	tipo desconocido
`varchar`	cadena de caracteres sin espacios al final, longitud especificada al momento de creación
`xid`	identificador de transacción

Los más importantes de todos estos son "`varchar`", "`integer`" y "`float`":

- **`varchar`**: se usa para almacenar cadenas de caracteres. Una cadena es una secuencia de caracteres. Se coloca entre comillas simples ('). El tipo "varchar" define una cadena de longitud variable en la cual determinamos el máximo de caracteres entre paréntesis. Puede guardar hasta 10485760 caracteres. Por ejemplo, para almacenar cadenas de hasta 30 caracteres, definimos un campo de tipo varchar(30), es decir, entre paréntesis, junto al nombre del campo colocamos la longitud. Si asignamos una cadena de caracteres de mayor longitud que la definida, la cadena no se carga, aparece un mensaje indicando tal situación y la sentencia no se ejecuta (`ERROR: value too long for type character varying(30)`).

- **`integer`**: se usa para guardar valores numéricos enteros, de -2000000000 a 2000000000 aprox. Definimos campos de este tipo cuando queremos representar, por ejemplo, cantidades.

- **`float`**: se usa para almacenar valores numéricos con decimales. Se utiliza como separador el punto (.). Definimos campos de este tipo para precios, por ejemplo.

pg_hba.conf y postgresql.conf

Archivos en la localización `/etc/postgresql/10/main/.`: (el 10 es por la versión)

- `pg_hba.conf`
- `postgresql.conf`
- `environment`
- `pg_ctl.conf`
- `pg_ident.conf`
- `start.conf`

```
root@paula-pc:/etc/postgresql/10/main# ls
environment  pg_hba.conf    postgresql.conf
pg_ctl.conf  pg_ident.conf  start.conf
```

De todos estos, me voy a centrar en dos de ellos, `pg_hba.conf` y `postgresql.conf`.

pg_hba.conf

La autentificación del cliente está controlada por un archivo de configuración, que normalmente se llama `pg_hba.conf` y está alojado en el directorio que anteriormente hemos nombrado. Cuando el directorio de datos e inicializado con `initdb`, se instala un `pg_hba.conf` predeterminado. Aun así, todos estos parámetros de configuración se pueden cambiar.

```
# DO NOT DISABLE!
# If you change this first entry you will need to make sure that the
# database superuser can access the database using some other method.
# Noninteractive access to all databases is required during automatic
# maintenance (custom daily cronjobs, replication, and similar tasks).
#
# Database administrative login by Unix domain socket
local   all             postgres                                peer

# TYPE  DATABASE        USER            ADDRESS                 METHOD

# "local" is for Unix domain socket connections only
local   all             all                                     peer
# IPv4 local connections:
host    all             all             127.0.0.1/32            md5
# IPv6 local connections:
host    all             all             ::1/128                 md5
# Allow replication connections from localhost, by a user with the
# replication privilege.
#local   replication     postgres                                peer
#host    replication     postgres        127.0.0.1/32            md5
```

Esta imagen es el archivo pg_hba.conf por defecto.

El formato general del archivo `pg_hba.conf` es una serie de registros, uno por línea. Las líneas en blanco son ignoradas al igual que si hay un texto tras el carácter de comentario (`#`). Los registros no pueden ser continuados a lo largo de las líneas, tienen que estar en líneas distintas. Un registro está hecho de un número de campos que están separados por espacios y/o tabulaciones. Los campos que pueden contener espacios en blanco si el valor del campo está entrecomillado.

Un registro puede tener uno de estos siete formatos:

```
local      database    user  auth-method [auth-options]
host       database    user  address     auth-method [auth-options]
hostssl    database    user  address     auth-method [auth-options]
hostnossl  database    user  address     auth-method [auth-options]
host       database    user  IP-address  IP-mask     auth-method [auth-options]
hostssl    database    user  IP-address  IP-mask     auth-method [auth-options]
hostnossl  database    user  IP-address  IP-mask     auth-method [auth-options]
```

Un ejemplo de error que puede salirnos al no tener este archivo bien configurado es este:

```
psql: FATAL: Peer authentication failed for user “postgres” (or any user)
```

Este error aparece al intentar iniciar sesión en PostgreSQL. Lo que hay que hacer es cambiar una línea del archivo `pg_hba.conf`.

De

```
local      all        all        peer
```

a

```
local      all        all        md5
```

Teniendo en cuenta que los tipos de autenticidad más importantes son estos:

- **`peer`**: va a confiar en la identidad (autenticidad) del usuario UNIX. Así que no va a pregunta por una contraseña.
- **`md5`**: siempre va a preguntar por una contraseña, y la validará después del hashing con MD5.
- **`trust`**: nunca va a preguntar por una contraseña y siempre va a confiar cualquier conexión.

En mi caso, ya enseñé anteriormente cómo se ve mi archivo `pg_hba.conf`:

```
local           all             postgres                peer

# TYPE  DATABASE        USER            ADDRESS                 METHOD
local           replication     postgres                peer

# "local" is for Unix domain socket connections only
#local   all             all                                     peer
local           all             all                     md5

# IPv4 local connections:
host            all             all             127.0.0.1/32            md5

# IPv6 local connections:
host            all             all             ::1/128                 md5
# Allow replication connections from localhost, by a user with the
# replication privileges
host            all             all             192.168.2.0/24          md5
```

postgresql.conf

Por otro lado, en el archivo `postgresql.conf` hay que cambiar tan solo una línea que normalmente aparece comentada. Habría que quitarle el # del comentario.
En esta línea pondrá "`listen_addresses='*'`". Lo que conseguimos con esto es que se escuchen todas las IP, en caso contrario ignoraría todas, incluida la nuestra y no podríamos conectarnos.

Comandos básicos Script

`ls`
Este comando lista los ficheros y directorios (en GNU/Linux todo es un fichero)
`ls /home`
Si queremos que nos muestre la información ampliada y en columnas, utilizaremos el parámetro '`-l`', y si además queremos que nos muestre los elementos ocultos, usaremos `ls -la`
Si queremos mostrar el número de archivos que hay en un directorio:
`ls | wc -l`

`cd`
Con este comando nos podemos mover entre diferentes directorios. Si nos queremos ir a un directorio el particular:
`cd /var/cache`
Si en cambio nos encontramos en un directorio, y queremos pasar al inmediatamente superior:
`cd ..`
O bien ir directamente a otro directorio que esta al mismo nivel que el nuestro:
`cd ../log`

`mkdir`
Con este comando podemos crear los que deseemos.
`mkdir directorio`
`mkdir directorio/subdirectorio`

`rmdir`
Podemos borrar un directorio:
`rmdir /directorio/subdirectorio`

`touch`
Con este comando podemos crear ficheros:
`touch fichero1 fichero2 fichero3`
Estos ficheros se encuentran vacíos.

`rm`
Al igual que con `touch` podemos crear nuevos ficheros, con `rm` podemos eliminarlos.
Mientras que `rmdir` eliminaba directorios, en este caso borra ficheros.
`rm ejemplo.txt`
Es un comando que pregunta con cada archivo, si quieres que se dedique a eliminar sin mala conciencia, sólo debemos añadir el parámetro `-f`
Si además quieres que elimine ficheros y subdirectorios, con `-r`.
`rm -rf` es la mezcla de estos.

`mv`

Este comando se usa para mover un fichero o directorio de lugar. También sirve para cambiar el nombre de un fichero.

```
mv /directorio/subdirectorio1/ejemplo.txt /directorio/subdirectorio2/
```

Y para cambiar el nombre:

```
mv /directorio/ejemplo.txt /directorio/nuevoejemplo.txt
```

`rename`

Cambia el nombre de un fichero o conjunto de ficheros. Tiene bastantes parámetros interesantes.

```
rename 's/.jpeg/.jpg/' *
```

De esta manera indicamos que en la ubicación se cambiarán todos los ficheros con extensión "`jpeg`" por "`jpg`"

`man`

Con este comando podemos consultar el manual, de ahí que se llame "`man`"

Si lo utilizamos seguido del comando a consultar, nos mostrará su entrada en el manual.

```
man touch
```

`info`

Es un comando similar al de `man` con información ampliada sobre el comando a consultar.

```
info touch
```

`clear`

Este comando se encarga de borrar la pantalla. Para utilizar solo hemos de escribir `clear`.

`sudo`

Con este comando nos podemos otorgar los poderes de super usuario, siempre que tengamos permisos para ello.

```
sudo su
```

`pwd`

Nos muestra el nombre del directorio de trabajo actual.

Un ejemplo:

```
[usuario@pc ~]$ pwd
/home/usuario
```

`cat`

Muestra el contenido de un fichero dado. Si se utiliza con varios ficheros a la vez, mostrará su contenido de manera secuencial.

```
cat ejemplo.txt
```

`chown`

Cambia el usuario y grupo propietarios de ficheros

```
chown usuario:familia fotos.gz
```

`find`
Busca un directorio o fichero específicos en el sistema de ficheros. Tiene una larga lista de opciones.

`locate`
Similar al comando '`find`', se encarga de buscar en el todo el sistema, ficheros o directorios que coincidan con una consulta. Por forma predeterminada busca únicamente en los ficheros que tiene permisos. A diferencia de '`find`' tiene su propia base de datos de consulta.

`wget`
Descarga el fichero o página web dada, indicando la URL
```
wget https://www.google.es/
```

`grep`
Busca en uno o más ficheros una cadena determinada de texto. Si encuentra la cadena nos indica donde está. Es un comando muy potente, muy utilizado por DevOps y desarrolladores.
```
cat /etc/passwd | grep -i mypass
```
Utilizando el parámetro '`-i`' ignoramos la diferencia entre mayúsculas y minúsculas.

`tail`
Imprime las diez últimas líneas de un fichero.
Es muy utilizado en la consulta de ficheros de registro.
```
tail -f -n 20 ejemplo.txt
```
De esta manera lista las últimas líneas de registro del fichero `ejemplo.txt`, con el parámetro "`-f`", indicamos que queremos ver la actividad del registro "en directo", con "`-n`" indicamos que queremos ver siempre las 20 últimas líneas.

`head`
Al igual que `tail` nos mostraba por defecto las diez últimas líneas, con head se nos muestran las diez primeras líneas.

`dpkg`
Es la herramienta habitual para trabajar con los paquetes con extensión `.DEB`, esto es, el gestor de paquetes de Debian. Su fin es instalar, compilar, eliminar y manipular los paquetes de Debian.

`df`
Nos informa de la utilización de disco en un sistema de ficheros. Este comando se encarga de mostrar el espacio usado y del disponible en todos los sistemas de ficheros montados.
Uno de los parámetros más usados es '`-h`' que muestra información más entendible.
```
df -h
```

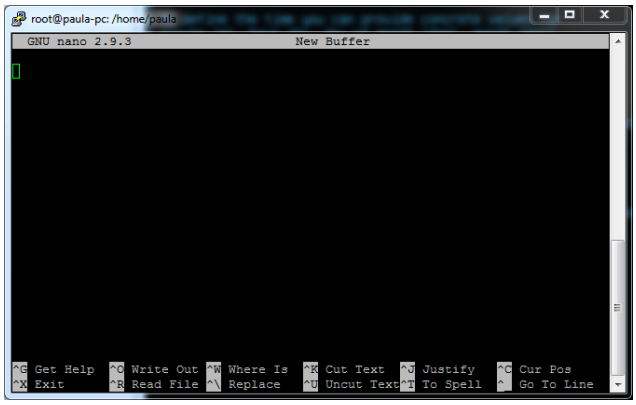
`mount`
`umount`
Se encarga de montar o desmontar un dispositivo dado sobre el sistema de ficheros.
```
mount recursonfs:/carpeta /carpeta
umount /carpeta
```

`sort`
Ordena el resultado de un texto dado

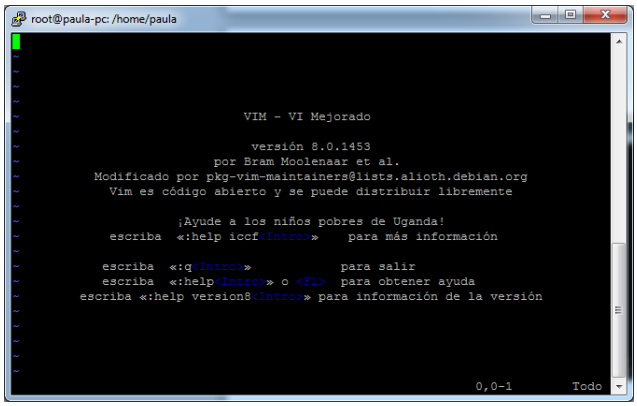
`more`
Se encarga de paginar texto, mostrando una pantalla cada vez.

`less`
Se trata de un programa similar a "`more`", pero más avanzado, ya que permite consultar páginas hacia atrás y hacia delante.

`nano archivo`
`vim archivo`
Estos dos comandos abren bien el editor de texto Nano o bien el editor de texto `Vim` con el archivo que se especifique. Estos dos editores hacen lo mismo pero con la diferencia de que en Nano aparecen abajo los comandos necesarios para utilizar el editor y en `Vim` tan solo aparecen al principio y después desaparecen.



Nano



Vim

chmod

Se encarga de cambiar los permisos de acceso a los ficheros.
A este comando le vamos a dar más importancia para entenderlo de forma adecuada.

> **Recuerda**: cuando queramos que un archivo sea ejecutable (como un script `.sh`) debemos otorgarle derechos de ejecución y eso se hace gracias a este comando:
>
> `chmod +x archivo.sh`

Cada archivo o directorio tiene tres grupos de permisos basados en usuario:

- Propietario: Los permisos de propietario solo aplican al propietario del archivo o directorio, no afectarán a las acciones de otros usuarios.
- Grupo: Los permisos de grupo se aplican solo al grupo que se ha asignado al archivo o directorio, no afectarán las acciones de otros usuarios.
- Todos los usuarios: Los permisos de "Todos los usuarios" se aplican a todos los demás usuarios del sistema, este es el grupo que más tenemos que vigilar.

Cada archivo o directorio tiene tres tipos de permisos básicos:

- Lectura: El permiso de lectura se refiere a la capacidad del usuario para leer el contenido del fichero.
- Escritura: Los permisos de escritura hacen referencia a la capacidad de un usuario para escribir o modificar un archivo o directorio.
- Ejecución: El permiso de ejecución afecta a la capacidad del usuario para ejecutar un archivo o ver el contenido de un directorio.

Visualización de permisos

Podemos ver los permisos verificando la información de los ficheros o directorios. Desde la terminal, podemos utilizar el comando "`ls -la`" o bien "`ll`"
La información mostrada será: "`_rwxrwxrwx 1 propietario:grupo`"
Más adelante aclararemos el significado de esto.

Derechos de usuarios y permisos

- El primer carácter que hemos marcado en el párrafo anterior con un guion bajo es el distintivo de permiso especial, que puede variar.
- El siguiente conjunto de tres caracteres (`rwx`) es para los permisos del propietario del fichero o carpeta.
- El segundo conjunto de tres caracteres (`rwx`) es para los permisos de grupo.
- Por último, el tercer conjunto de tres caracteres (`rwx`) es para los permisos del resto de usuarios del sistema.
- La última parte nos muestra la asignación de propietario y grupo formateada como "`Propietario:Grupo`"

Definir los permisos de manera explícita

Para definir de esta manera los permisos, necesitaremos hacer referencia al grupo de permisos y tipos de permisos.
Los grupos de permisos utilizando son:

- `u` – usuario propietario
- `g` – grupo
- – otros
- `a` – todos los usuarios

Además utilizamos los operadores + (más) y – (menos) ; estos se utilizan para decirle al sistema si agrega o elimina los permisos específicos.
Los tipos de permisos que se usan son:

- `r` – leer
- `w` – escribir
- `x` – ejecutar

Ejemplos:
"`fichero1`" tiene permisos "`_rw_rw_rw`", lo que significa que el propietario, el grupo y todos los usuarios tienen permisos de lectura y escritura, pero no de ejecución.
Vamos a eliminar los permisos de lectura y escritura de todos los grupos de usuarios.

```
chmod a -rw fichero
```

Para agregar otra vez los permisos, los haríamos así:

```
chmod a +rw fichero
```

Definir los permisos utilizando referencias binarias

Una cadena de permiso de muestra sería "`chmod 640 fichero1`", lo que significa que el propietario tiene permisos de lectura y escritura, el grupo tiene permisos de lectura y el resto de los usuarios no tienen ningún tipo de derecho sobre el fichero.
El primero dígito representa el permiso de propietario; el segundo representa los permisos del grupo y por último el último número representa los permisos para todos los demás usuarios.
Los números son una representación binaria de la cadena "`rwx`", de la que ya hemos hablado antes:

- `r = 4` (2^2)
- `w = 2` (2^1)
- `x = 1` (2^0)

Los números se suman o se restan para obtener el número entero que va a representar los permisos que deseamos establecer. Deberemos incluir los permisos binarios para cada u no de los tres grupos de permisos.

Ejemplo: vamos a otorgar todos los permisos al usuario propietario, incluido el de ejecución; al grupo le daremos permisos de lectura y ejecución; y al resto de usuarios no les daremos ningún permiso, sería así:

```
chmod 750 fichero
```

Permisos especiales

El indicador de permiso especial se puede marcar con cualquier de los siguientes:

- `_` – sin permisos especiales
- `d` – directorio
- `l` – El archivo o directorio es un enlace simbólico
- `s` – Esto indica los permisos "`setuid / setgid`". Esto no se establece en la parte de permisos especiales de la pantalla de permisos, sino que se representa como una s en la parte de lectura de los permisos de propietario o grupo.
- `t` – Esto indica los permisos de bit adhesivo. Esto no se establece en la parte de permiso especial de la pantalla de permisos, sino que se representa como una t en la parte ejecutable de todos los permisos de los usuarios.

Los permisos "`setuid / setgid`" se utilizan para indicar al sistema que ejecute un ejecutable como el propietario con los permisos de propietario.

Tenemos que tener cuidado al utilizar "`setuid / setgid`" en los permisos. Si asignamos permisos incorrectamente a un archivo propiedad del usuario root con "`setguid / setgid bit set`" podemos abrir el sistema a posibles intrusiones.
A modo de resumen, veamos alguno ejemplo de lo visto hasta ahora:

Cadena de permisos	Código octal	Significado
`rwxrwxrwx`	777	Permisos de lectura, escritura y ejecución para todos los usuarios.
`rwxr-xr-x`	755	Permisos de lectura y escritura para todos los usuarios. El propietario del fichero también tiene permisos de lectura.
`rwxr-x---`	750	Permisos de lectura y ejecución para el propietario y el grupo. El propietario del fichero también posee permiso de escritura. Los usuarios que no son propietarios del fichero o miembros del grupo no tienen acceso a éste.
`rwx------`	700	Permisos de lectura, escritura y ejecución sólo para el propietario del fichero; los demás no tendrán acceso.
`rw-rw----`	660	Permisos de lectura y escritura para el propietario y el grupo. No hay permisos globales.
`rw-r--r--`	644	Permisos de lectura y escritura para el propietario. Permiso de sólo lectura para el resto.
`rw-------`	600	Permisos de lectura y escritura para el propietario. Nadie más tiene permisos.

Heredar permisos

Si nos interesa que todos los ficheros y directorios que estén por debajo de un directorio en concreto, tengan los mismos permisos que el directorio "madre", lo lograremos añadiendo al comando `-R`, es decir, de manera recursiva:

```
chmod 750 carpeta1 -R
```

Nota: Voy a hablar también de forma separada a lo anterior de los comandos para archivos comprimidos y de los comandos utilizados en redes.

Comandos de archivos comprimidos

En cuanto a comandos de archivos comprimidos voy a hablar de cuatro tipos de archivos: ZIP, TAR, TAR.GZ y 7z, los más usados

ZIP

Es el más famoso de todos y lleva mucho años en funcionamiento. Funciona en todos los sistemas operativos (Windows, MAC, Unix y GNU/Linux). ZIP forma parte del primer grupo de los formatos de compresión:

- Es rápido al comprimir.
- Utiliza pocos recursos del sistema.
- Su ratio de compresión es pobre.

Su funcionamiento básico es el siguiente:

```
zip file1 aguardar1 aguardar2 aguardar3
```

Ejecutando el comando, crearemos el archivo `file1.zip` donde tendremos comprimidos los archivos: `aguardar1 aguardar2 aguardar3`.

```
zip -r file1 .
```

El comando comprimirá todo el contenido, incluyendo subcarpetas del directorio actual en el archivo `file1.zip`

```
zip -r file1 carpeta1
```

Comprimirá toda la carpeta1 en al archivo de comprimido `file1.zip`

TAR

Es un formato muy usado en entornos UNIX y GNU/Linux:

- Es muy rápido realizando su trabajo de compresión.
- Utiliza muy pocos recursos de la máquina.
- El nivel de compresión es bajo.

Algunos ejemplos de uso serían:
`tar -cwf ficherobackup /folder`
Lo que hacemos es indicar al comando `tar` que nos comprima la carpeta folder y todo su contenido en el archivo `ficherobackup.tar`

Parámetros:

- `-c` o `--create`, es decir, creamos un archivo o contenedor.
- `-w` o `--interactive`, interactivo con confirmación.
- `-f` o `--file`, especifica el nombre del contenedor.

Ahora vamos a descomprimir.
`tar -xwf comprimido.tar`
Extraemos el contenido del archivo comprimido.tar

Parámetros:

- `-x`, extraemos.

El resto de parámetros están explicados en el ejemplo anterior.

Veamos otro caso:
`tar -xvf archivo.tar /tmp/descargas/`
Ahora descomprimimos `archivo.tar` en el directorio `/tmp/descargas`.

Parámetros:

- `-v` o `--verbose`, es decir, queremos que nos dé más detalle de la descompresión.

TAR.GZ

Aquí tenemos un formato que es:

- Un punto intermedio entre el grado de compresión y la utilización de los recursos.
- Muy usado en los departamentos de sistemas.

Parámetros:

- `-c`, crea un archivo.
- `-z`, utiliza el formato `gzip`
- `-f`, indicamos que el siguiente argumentos es el fichero comprimido `.tar.gz`
- `-v`, al igual que con TAR nos dará más detalle sobre la compresión.

Un ejemplo de compresión:
`tar -czfv file.tar.gz ficheros`
Con el comando comprimiremos los ficheros en el archivo `file.tar.gz`

Parámetros:

- `-x`, al igual que con TAR indicamos que vamos a extraer.
- `-z`, utiliza el formato gzip
- `-v`, detalles de la descompresión.
- `-f`, indicamos que el siguiente argumento, es el fichero a descomprimir.

Ejemplo:
`tar -xzvf fichero.tar.gz`

7z

- Es un formato de compresión libre.
- A diferencia de ZIP tiene un ratio de compresión más alto.

El funcionamiento:
`7z a file.7z folder`
Con el parámetro 'a' indicamos que vamos a comprimir el directorio folder en el fichero `file.7z`

`7z e file.7z`
Con el parámetro 'e' extraemos el contenido del fichero `file.7z` en la ubicación actual.

> **Nota:** hay que decir que el formato 7z no viene por defecto en muchas distribuciones.

Hay varios programas que tratan con éste formato. Los podemos instalar vía consola de comandos:
`sudo apt-get install p7zip`

Comandos de administración de redes

`Ping`

Quizás la más básica e importante herramienta en lo que respecta a las redes.

Ping se encarga de verificar la conectividad de extremo a extremo, desde nuestro sistema hacia el que queremos conectar.

Para IPv4: `ping < dirección IP o nombre DNS >`

Para IPv6: `ping6 < dirección IP o nombre DNS >`

`Traceroute`

Con esta herramienta podemos rastrear la ruta completa de red, desde nuestro sistema al de un tercero. Cuando ping comprueba la conectividad de extremo a extremo, la utilidad traceroute le dice a todas las IP del enrutador por las que pasa, hasta llegar a su destino.

Es una herramienta habitual para comprobar el punto de fallo en una conexión.

`Traceroute <dirección IP o nombre DNS >`

`Telnet`

Se trata de un protocolo de red, que nos permite acceder a otra máquina para manejarla de manera remota. También es el nombre del programa informático, que es nuestro caso, que implementa el cliente.

`telnet < dirección IP o nombre DNS >`

También podemos utilizar la herramienta para saber si la máquina que queremos consultar tiene un puerto abierto o cerrado:

`telnet < dirección IP o nombre DNS >  < puerto >`

`Iptables`

Es sin duda la herramienta principal para gestionar el cortafuegos de los servidores o equipos domésticos. Permite filtrar, bloquear o permitir cierto tráfico. Es una utilidad muy potente.

`Lsof`

Se encarga de listar los puertos abiertos en el servidor.

Cron

Los cronjobs, tal y como indica su nombre son tareas cronológicas, es decir, tareas preprogramadas que se realizan cada cierto tiempo. Un uso muy importante son las copias de seguridad.
La sintaxis:

```
1 2 3 4 5 /path/to/command arg1 arg2
```

O

```
1 2 3 4 5 /root/backup.sh
```

```
* * * * * comando a ejecutar
- - - - -
| | | | |
| | | | ----- Dia de la semana (0 - 7) (Domingo=0 o 7)
| | | ------- Mes (1 - 12)
| | --------- Dia del mes (1 - 31)
| ----------- Hora (0 - 23)
------------- Minuto (0 - 59)
```

Para editar o crear tu propio archivo cron:

```
$ crontab -e
```

Operadores:

- * : todos los valores posibles para un campo. Por ejemplo, en el campo de la hora equivale a "cada hora".
- , : una lista de valores, por ejemplo: "1,5,10,15,20, 25".
- - : un rango de valores, por ejemplo: "5-15" equivale a "5,6,7,8,9,...,13,14,15".
- / : valores por etapa, por ejemplo: "0-23/" se puede utilizar en el campo de horas para especificar que se ejecute cada hora. Las etapas también son permitidas después de un asterisco, así que si quieres decir "cada dos horas" debes utilizar */2.

Lista todos tus cronjobs:

```
# crontab -l
# crontab -u username -l
```

Para eliminar todos tus cronjobs:

```
crontab -r
```

Nota: se elimina el hecho de que sean cron, no el archivo en sí.

Special string	Meaning
`@reboot`	Se ejecuta una vez, al encendido.
`@yearly`	Se ejecuta una vez al año, “0 0 1 1 *”.
`@annually`	(igual que `@yearly`)
`@monthly`	Se ejecuta una vez al mes, “0 0 1 * *”.
`@weekly`	Se ejecuta una vez a la semana, “0 0 * * 0”.
`@daily`	Se ejecuta una vez al día, “0 0 * * *”.
`@midnight`	(igual que `@daily`)
`@hourly`	Se ejecuta una vez cada hora, “0 * * * *”.

En mi caso, he puesto que los scripts se ejecuten automáticamente todos los días de la semana a las 9:30 (la copia de seguridad) y 9:40 (la subida al drive)

```
$ crontab -e

30 9 * * * /home/paula/cron/backup.sh
40 9 * * * /home/paula/cron/uplodrive.sh
```

Nota: hablaré de estos archivos posteriormente.

Drive

De cómo he conseguido que se suba todo al drive, voy a hablar a continuación:

```
$ sudo add-apt-repository ppa:twodopeshaggy/drive
$ sudo apt-get update
$ sudo apt-get install drive
$ go get github.com/odeke-em/drive/cmd/drive
```

Para poder usar drive es necesario inicializar la carpeta que sincronizaremos con nuestra cuenta.

Primero hay que crear la carpeta y movernos hacia ella.
```
mkdir drive
cd drive
```

Ahora inicializamos el directorio con el siguiente comando:
```
$ drive init
```

Esto generará un enlace para permitir que drive acceda a tu cuenta de google.
Al abrirlo en tu navegador te pedirá que inicies sesión en tu cuenta, posteriormente generará un código de autorización que tendrás que copiar para pegarlo en el terminal.

Comandos de drive útiles

- `cp` – copiar caminos de directorios remotos al destino.
- `del` – elimina los archivos permanentemente. Esta operación es irreversible.
- `diff` – compara los archivos locales con sus remotos equivalentes.
- `mv` – mueve archivos/directorios.
- `open` – abre un archivo en el administrador de archivos apropiado o en el navegador predeterminado.
- `pull` – extrae cambios remotos desde Google Drive.
- `push` – envía cambios locales a Google Drive.
- `rename` – renombra un archivo/directorio.
- `share` – comparte archivos con correos email específicos otorgando a los usuarios especificados roles y permisos.
- `trash` – mueve archivos a la papelera.
- `untrash` – restaura archivos desde la papelera a sus localizaciones originales.

Scripts para hacer copia de seguridad y subirla a drive

Visto todo lo anterior, voy a hablar de dos scripts que sirven para hacerle copias de seguridad diarias a la base de datos que creé y para subir esta copia de seguridad a drive. Las copias de seguridad se suben en tres directorios llamados copia1, copia2 y copia3 que contienen en copia1 la copia de seguridad más reciente, en copia2 la anterior y en copia3 la anterior a esta. Las copias de seguridad se suben con la fecha y hora del momento en el que se realizan.

Script para hacer una copia de seguridad

El archivo se encuentra en el directorio `/home/paula/cron/backup.sh`

```
#!/bin/bash

backups_path="/home/paula/cron/backups"
current_date_time="$(date +%Y_%m_%d-%H:%M:%S)"

pg_dump -h 192.168.2.194 -U paula –f $backups_path/"$current_date_time".backup mi_db

rm -rf $backups_path/copia3/*
cp $backups_path/copia2/* $backups_path/copia3/

rm -rf $backups_path/copia2/*
cp $backups_path/copia1/* $backups_path/copia2

rm -rf $backups_path/copia1/*
mv $backups_path/"$current_date_time".backup $backups_path/copia1
```

Paso a paso

La primera línea de nuestro script le indica al sistema que tiene que usar la shell BASH.
Todas las líneas que comiencen por # son ignoradas por la máquina y nos sirven para incluir comentarios destinados a programadores o usuarios excepto esta primera.

```
#!/bin/bash
```

Creamos dos variables, la primera contiene la ruta en la que se encuentran los tres directorios copia1, copia2 y copia3. Esto lo hacemos para no tener que escribir cada vez la ruta sino directamente la variable.
La segunda variable es la que contiene la hora exacta en que se está haciendo la copia de seguridad. Esto lo hacemos para ponerle el nombre de la fecha a las copias de seguridad. De esta forma sabremos qué fecha y hora le corresponde a cada copia de seguridad.

```
backups_path="/home/paula/cron/backups"
current_date_time="$(date +%Y_%m_%d-%H:%M:%S)"
```

Este comando es el que hace la copia de seguridad y la guarda (-f) en el directorio, con el nombre de la fecha y la hora y la extensión ".backup" para diferenciarlo del resto de archivos. Esto no va a perjudicar en nada, la copia de seguridad seguirá siendo un archivo de texto.

```
pg_dump -h 192.168.2.194 -U paula –f $backups_path/"$current_date_time".backup mi_db
```

Se borra lo que había en copia3 y se pasa la copia de seguridad de copia2 a copia3.

```
rm -rf $backups_path/copia3/*
cp $backups_path/copia2/* $backups_path/copia3/
```

Se borra lo que había en copia2 y se pasa la copia de seguridad de copia1 a copia2.

```
rm -rf $backups_path/copia2/*
cp $backups_path/copia1/* $backups_path/copia2
```

Se borra lo que había en copia1 y se pasa la copia de seguridad que acabamos de hacer que está en la carpeta backups (la carpeta madre de copia1, copia2 y copia3) al directorio copia1.

```
rm -rf $backups_path/copia1/*
mv $backups_path/"$current_date_time".backup $backups_path/copia1
```

Script para subir la copia de seguridad al drive

Este archivo se encuentra en el directorio `/home/paula/cron/uplodrive.sh` .

```
#!/bin/bash

cp -r /home/paula/cron/backups/* /home/paula/drive

rdfind /home/paula/drive/
rm results.txt

cd /home/paula/drive
drive push

rm -r /home/paula/drive/*
```

Paso a paso.

Esta primera línea es igual a la del anterior script, le indica al sistema que tiene que usar la shell BASH.

```
#!/bin/bash
```

Copiamos todo lo que había en la carpeta backups donde anteriormente hemos hecho la copia de seguridad y lo copiamos a la carpeta donde hemos iniciado sesión de drive

```
cp -r /home/paula/cron/backups/* /home/paula/drive
```

Este programa lo he instalado anteriormente en el sistema, busca archivos que sean iguales en el directorio. A su salida realiza un archivo txt llamado results.txt que elimino cuando se ha ejecutado.

```
rdfind /home/paula/drive/
rm results.txt
```

Me muevo al directorio de drive y ejecuto esta orden que lo que hace es que envía todo lo de la carpeta a Google Drive.

```
cd /home/paula/drive
drive push
```

Elimina todo lo que hay en la carpeta drive para que después no haya problemas, ya está todo a salvo en la nube.

```
rm -r /home/paula/drive/*
```

Configuración inicial de un servidor Linux

A continuación, voy a hablar sobre la configuración inicial de un servidor Linux, la instalación de una instancia LAMP (Linux, Apache, MySQL y PHP) y la instalación de Wordpress junto con LAMP.

Cuando se crea un nuevo servidor de Ubuntu, hay algunos pasos de configuración que se deben seguir desde el principio como parte de la configuración básica.

> **Estos pasos son**: iniciar sesión de root, crear una cuenta de nueva usuario con un reducido margen de influencia, privilegios de root para la cuenta creada, registro de prueba con este usuario, configurar un firewall básico que permita conexiones SSH.

Paso 1: Sesión de root

Para iniciar sesión en el servidor, necesitarás conocer la dirección IP pública del servidor.
También necesitarás la contraseña para la cuenta "root" del usuario.

Si no estás conectado a tu servidor, inicia sesión como usuario root.

> **Acerca del usuario root**: el usuario root es el usuario de administración en un entorno Linux que tiene muy amplios privilegios. Debido a los privilegios elevados de la cuenta root, en realidad se desaconseja usarlo de forma regular. Esto se debe en parte del poder inherente a la cuenta de root es la capacidad de hacer cambios muy destructivos, incluso por accidente.

Paso 2: Crear una cuenta de usuario alternativa con un reducido margen de influencia para el trabajo del día a día

Una vez que se ha iniciado sesión como root, estamos preparados para agregar la nueva cuenta de usuario que usaremos para iniciar sesión de ahora en adelante.
En este ejemplo se crea un nuevo usuario llamado "`paula`", pero puedes reemplazarlo por el nombre de usuario que quieras:

```
adduser paula
```

Se te harán algunas preguntas, comenzando con la contraseña de la cuenta. Puedes completar la información adicional si lo deseas. Esto no es necesario y puedes tan solo ir pulsando ENTER en los campos que desees omitir (la contraseña no es opcional).

Paso 3: Privilegios de root

Ahora, tenemos una nueva cuenta de usuario con privilegios de cuenta regulares. Sin embargo, es posible que a veces tengas que realizar tareas administrativas (aquellas que necesitan privilegios de root).

Para evitar tener que cerrar la sesión de nuestro usuario normal y volver a iniciar sesión como la cuenta root, podemos establecer lo que se conoce como "superusuario" o privilegios de root para nuestra cuenta normal.
Esto permitirá a nuestro usuario normal ejecutar comandos con privilegios administrativos colocando la palabra sudo antes de cada comando.

Para añadir estos privilegios a nuestro nuevo usuario, tenemos que añadir el nuevo usuario al grupo "sudo". A los usuarios que pertenecen al grupo "sudo" se les permite usar el comando sudo.
Como root, ejecuta este comando para añadir el nuevo usuario al grupo sudo:

```
usermod -aG sudo paula
```

Ahora tu usuario puede ejecutar comandos con privilegios de superusuario.

Paso 4: Registro de prueba

Ahora, antes de salir del servidor, debes probar tu nueva configuración.
No desconectes hasta que confirmes que puedes iniciar sesión correctamente a través de SSH.

En una nueva terminal en tu máquina local, inicia sesión en tu servidor utilizando la nueva cuenta que creamos.
Para ello, utilice este comando:

```
ssh usuario@ip_del_servidor
```

En mi caso:

```
ssh paula@192.168.X.X
```

Se te pedirá la contraseña de tu usuario.

Una vez que se proporciona la autenticación al servidor, se registrará como tu nuevo usuario.

> **Nota**: si necesitas ejecutar un comando con privilegios de root, escribe "sudo" antes del comando.

Paso 5: Configurar un firewall básico

Los servidores Ubuntu 18.04 pueden usar el firewall UFW para asegurarse de que sólo se permiten conexiones a ciertos servicios.
Podemos configurar un firewall básico fácilmente utilizando estos comandos.

Distintas aplicaciones pueden registrar sus perfiles con UFW después de la instalación.
Estos perfiles permiten al UFW gestionar estas aplicaciones por su nombre.
OpenSSH, el servicio que nos permite conectarnos a nuestro servidor ahora, tiene un perfil registrado con UFW.

Puedes verlo escribiendo:

```
sudo ufw app list
```

A la salida:

```
Available applications:
  OpenSSH
```

Necesitamos asegurarnos de que el firewall permita conexiones SSH para que podamos volver a conectarnos la próxima vez.
Podemos permitir estas conexiones escribiendo:

```
sudo ufw allow OpenSSH
```

Posteriormente, podemos habilitar el firewall escribiendo:

```
sudo ufw enable
```

Escribe "y" y presiona ENTER para continuar.
Puedes ver que las conexiones SSH todavía se permiten escribiendo:

```
sudo ufw status
```

A la salida:

```
Status: active

To                         Action      From
--                         ------      ----
OpenSSH                    ALLOW       Anywhere
OpenSSH (v6)               ALLOW       Anywhere (v6)
```

Si instalas y configurar servicios adicionales, deberás ajustar la configuración del firewall para permitir el tráfico aceptable.

LAMP

Un "LAMP" stack es un grupo de software libre que se suele instalar junto para permitirle a un servidor tener distintas páginas web y aplicaciones.
Su nombre viene dado por las iniciales de los programas que tiene:

- Linux
- Apache
- MySQL: es similar a PostgreSQL
- PHP

Los datos de todo esto se almacenan en una base de datos en MySQL, el contenido se procesa en PHP.

Como ya sabemos, sistema operativo utilizado es Ubuntu Server 18. Con esto conseguimos la L de LAMP, un sistema operativo Linux.

A partir de aquí iremos viendo el resto de requisitos.

Paso 1: Instalar Apache

Al instalar Apache no podemos olvidarnos de que tendremos que permitir que pueda funcionar por el cortafuegos. Si no hacemos esto, no nos funcionará.
El servidor web Apache es el más popular a lo largo de todo el mundo. Está muy bien documentado, cuenta con numerosos foros y se ha ido utilizando por mucha gente que se ha dedicado al mundo web. Esto hace que sea una perfecta opción para tener ahí nuestra web.

La forma más sencilla de instalar Apache es utilizando `apt`:
Teniendo nuestro sistema operativo actualizado (`apt-get update && apt-get upgrade`), haremos:

```
sudo apt-get install apache2
```

Establecer `Global ServerName` para hacer que cesen las advertencias de sintaxis

Cuando escribimos el comando:

```
sudo apache2ctl configtest
```

Podemos detectar si existen errores en la configuración del Apache. Pero a la salida recibimos el siguiente mensaje:

```
AH00558: apache2: Could not reliably determine the server's fully
qualified domain name, using 127.0.1.1. Set the 'ServerName' directive
globally to suppress this message
Syntax OK
```

Este mensaje es una advertencia que no avisa de nada importante. Para hacer que no vuelva a salir, tendremos que editar el archivo `apache2.conf`:

```
sudo nano /etc/apache2/apache2.conf
```

Dentro de este archivo, al final, incluiremos una línea de texto:

```
ServerName dominio_del_servidor_o_IP
```

En mi caso, como no tengo dominio, pondré la IP.

```
ServerName 82.X.X.X
```

> **Nota**: podemos averiguar nuestra IP pública si nos metemos en la web cualesmiip.com u otra similar.

Ahora, al ejecutar el comando:

```
sudo apache2ctl configtest
```

Tendremos a la salida:

```
Syntax OK
```

Reiniciamos Apache para implementar los cambios:

```
sudo systemctl restart apache2
```

Cortafuegos

Ahora viene la segunda parte, ajustar el cortafuegos:

Si seguimos las instrucciones anteriores, al ejecutar el comando:

```
sudo ufw app list
```

Tendremos OpenSSH funcionando y tres más de Apache sin funcionar.

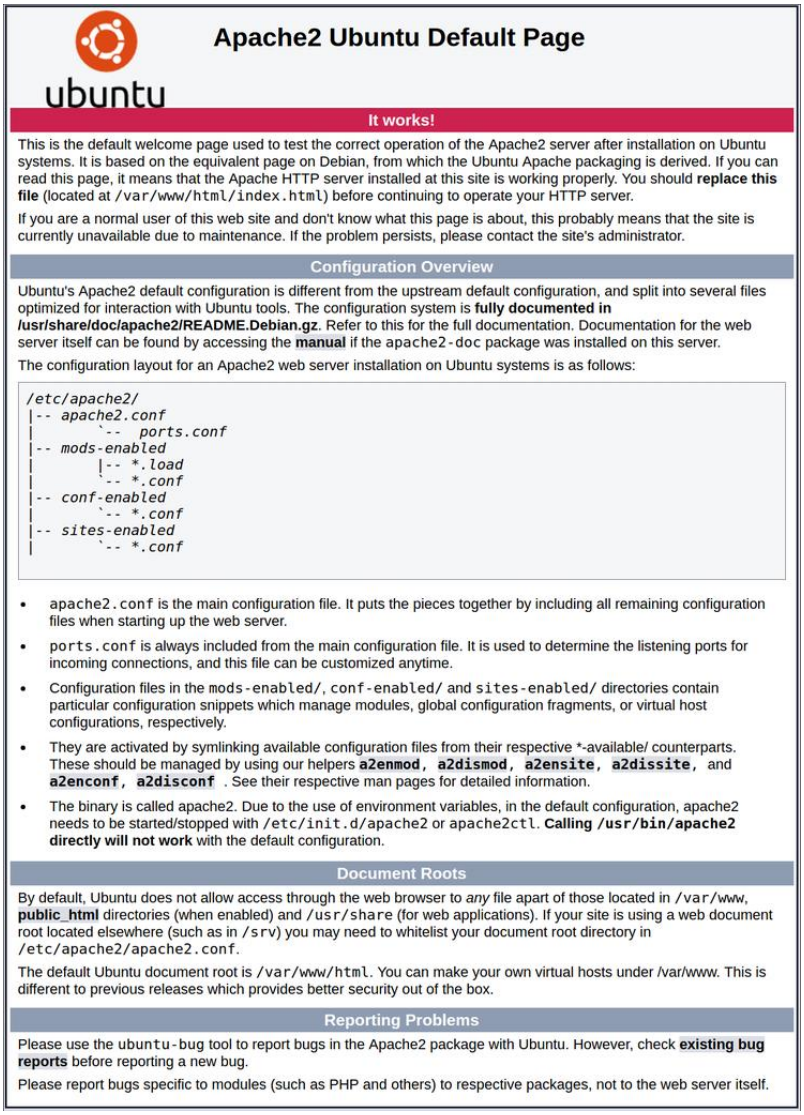
Vamos a permitir "`Apache Full`", esto hará que el cortafuegos permita el funcionamiento libre de Apache:

```
sudo ufw allow in "Apache Full"
```

Tras todo esto, podremos comprobar si todo funciona escribiendo en cualquier navegador:

```
http://IP_de_tu_servidor
```

Verás en el navegador la página predeterminada de Apache



Apache2 Ubuntu Default Page

ubuntu

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

Configuration Overview

Ubuntu's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Ubuntu tools. The configuration system is **fully documented in /usr/share/doc/apache2/README.Debian.gz**. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the `apache2-doc` package was installed on this server.

The configuration layout for an Apache2 web server installation on Ubuntu systems is as follows:

```
/etc/apache2/
|-- apache2.conf
|       `--  ports.conf
|-- mods-enabled
|       |-- *.load
|       `-- *.conf
|-- conf-enabled
|       `-- *.conf
|-- sites-enabled
|       `-- *.conf
```

- `apache2.conf` is the main configuration file. It puts the pieces together by including all remaining configuration files when starting up the web server.
- `ports.conf` is always included from the main configuration file. It is used to determine the listening ports for incoming connections, and this file can be customized anytime.
- Configuration files in the `mods-enabled/`, `conf-enabled/` and `sites-enabled/` directories contain particular configuration snippets which manage modules, global configuration fragments, or virtual host configurations, respectively.
- They are activated by symlinking available configuration files from their respective *-available/ counterparts. These should be managed by using our helpers **`a2enmod`**, **`a2dismod`**, **`a2ensite`**, **`a2dissite`**, and **`a2enconf`**, **`a2disconf`** . See their respective man pages for detailed information.
- The binary is called apache2. Due to the use of environment variables, in the default configuration, apache2 needs to be started/stopped with `/etc/init.d/apache2` or `apache2ctl`. **Calling `/usr/bin/apache2` directly will not work** with the default configuration.

Document Roots

By default, Ubuntu does not allow access through the web browser to *any* file apart of those located in `/var/www`, **public_html** directories (when enabled) and `/usr/share` (for web applications). If your site is using a web document root located elsewhere (such as in `/srv`) you may need to whitelist your document root directory in `/etc/apache2/apache2.conf`.

The default Ubuntu document root is `/var/www/html`. You can make your own virtual hosts under /var/www. This is different to previous releases which provides better security out of the box.

Reporting Problems

Please use the `ubuntu-bug` tool to report bugs in the Apache2 package with Ubuntu. However, check **existing bug reports** before reporting a new bug.

Please report bugs specific to modules (such as PHP and others) to respective packages, not to the web server itself.

Si te encuentras esta página, tu servidor está correctamente instalado y se puede acceder correctamente con el cortafuegos.

Paso 2: Instalar MySQL

Ahora que ya tenemos la A de Apache, toca ir a por MySQL.
Lo primero será instalarlo:

```
sudo apt-get install mysql-server
```

Durante la instalación, se te preguntará una contraseña para el usuario root de MySQL. Esta cuenta root es la cuenta administrativa que tiene todos los privilegios.
Cuando la instalación está completa, ejecutaremos un script para terminar de configurar MySQL:

```
mysql_secure_installation
```

Te pedirán que escribas la contraseña que elegiste para el usuario root y te dirán que elijas si ejecutar "`VALIDATE PASSWORD PLUGIN`".
Este plugin se utiliza para mejorar la seguridad y probar las contraseñas, si quieres hacer que la seguridad sea más elevada, acepta esto. Te pedirán cierta información. En mi caso, no quiero mejorar mi seguridad así que no voy a ejecutar el plugin.

Después de todo esto, te pedirán una nueva contraseña, puedes escribir la anterior o cambiarla.
A continuación, te harán algunas preguntas a las que debes contestar de forma afirmativa. Eso hará que se eliminen usuarios anónimos, la base de datos de prueba...

Paso 3: Instalar PHP

Esta ya es la última instalación de software, la P de LAMP.
PHP es el componente que se encarga de procesar el código. Funciona con scripts, conectado con MySQL a nuestras bases de datos y entregar el contenido procesado a nuestro servidor para mostrarlo.
A la hora de instalarlo, vamos a hacerlo incluyendo algunos paquetes que nos serán útiles más adelante ya que están preparados para trabajar con Apache y MySQL.

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

Con este comando se debería instalar PHP sin ningún problema. Aun así, lo comprobaremos más adelante.
En la mayoría de los casos, vamos a querer modificar el modo en el que Apache muestra los archivos cuando se solicita un directorio.
Actualmente, si un usuario solicita un directorio desde el servidor, Apache primero va a buscar un archivo llamado `index.html`. Queremos decirle que en nuestro caso preferimos los archivos PHP en vez de html, así que vamos a decirle que primero busque un archivo `index.php` en vez de `index.html`.

Para hacer esto, debemos abrir el archivo `dir.conf`. Como es un archivo de administración, lo tendremos que abrir obligatoriamente con derechos de superusuario.

```
sudo nano /etc/apache2/mods-enabled/dir.conf
```

Se verán estas líneas:

```
/etc/apache2/mods-enabled/dir.conf
<IfModule mod_dir.c>
    DirectoryIndex index.html index.cgi index.pl index.php index.xhtml
index.htm
</IfModule>
```

Cambiamos el orden por este otro:

```
/etc/apache2/mods-enabled/dir.conf
<IfModule mod_dir.c>
    DirectoryIndex index.php index.html index.cgi index.pl index.xhtml
index.htm
</IfModule>
```

Cuando acabes, guarda y cierra con Ctrl+X, confirmas el guardado con Y y pulsas ENTER para mantener el nombre del archivo y su ruta.

Tras esto, tenemos que resetear el servidor Apache para que se reconozcan los cambios que hemos hecho. Se hace escribiendo:

```
service apache2 restart
```

Podemos también ver el estado de Apache con:

```
service apache2 status
```

Este comando nos puede servir de ayuda si, por ejemplo, al entrar alguna vez desde el navegador a la IP nos da error. Podemos ver si Apache está funcionando o si está parado.

Paso 4: Probar el proceso de PHP en nuestro servidor web

Para comprobar si nuestro sistema está configurado adecuadamente para PHP, podemos crear un script muy básico al que llamaremos `info.php`.
Para que Apache lo encuentre tenemos que guardarlo en un directorio determinado llamado "web root".
En Ubuntu este directorio está en `/var/www/html/`. Podemos crear el archivo `info.php` así:

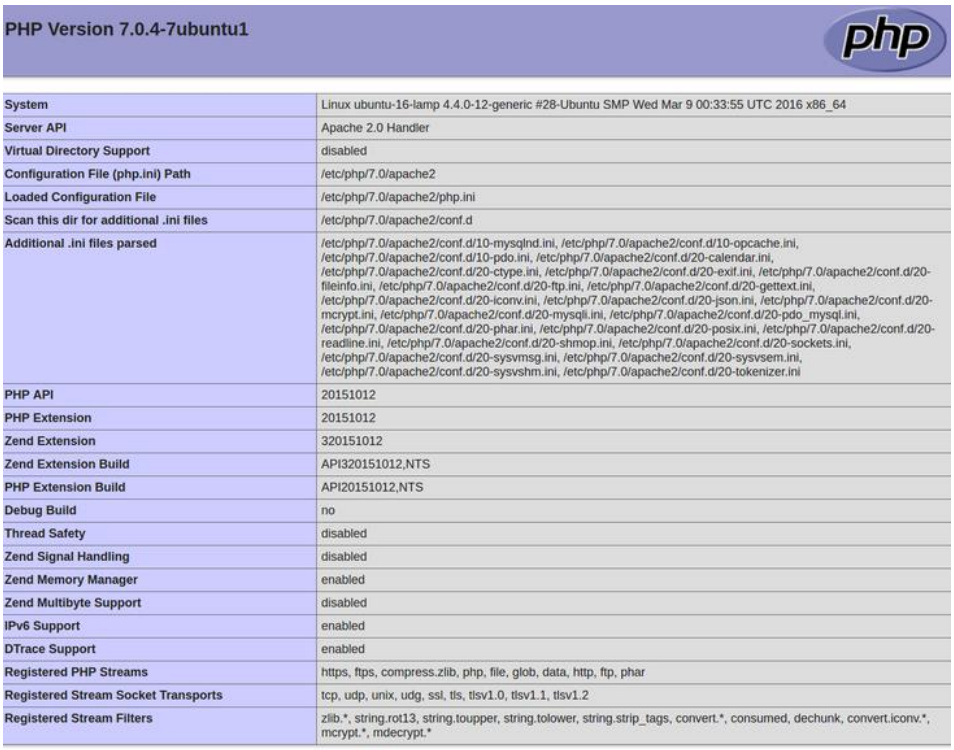
```
sudo nano /var/www/html/info.php
```

Esto abrirá un archivo vacío al que le vamos a meter el siguiente código en PHP:

```
<?php
phpinfo();
?>
```

Tras guardarlo, podremos comprobar el buen funcionamiento escribiendo en el buscador:
`http://la_ip_del_servidor/info.php`

La imagen que se verá será algo así:



PHP Version 7.0.4-7ubuntu1

System	Linux ubuntu-16-lamp 4.4.0-12-generic #28-Ubuntu SMP Wed Mar 9 00:33:55 UTC 2016 x86_64
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.0/apache2
Loaded Configuration File	/etc/php/7.0/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/7.0/apache2/conf.d
Additional .ini files parsed	/etc/php/7.0/apache2/conf.d/10-mysqlnd.ini, /etc/php/7.0/apache2/conf.d/10-opcache.ini, /etc/php/7.0/apache2/conf.d/10-pdo.ini, /etc/php/7.0/apache2/conf.d/20-calendar.ini, /etc/php/7.0/apache2/conf.d/20-ctype.ini, /etc/php/7.0/apache2/conf.d/20-exif.ini, /etc/php/7.0/apache2/conf.d/20-fileinfo.ini, /etc/php/7.0/apache2/conf.d/20-ftp.ini, /etc/php/7.0/apache2/conf.d/20-gettext.ini, /etc/php/7.0/apache2/conf.d/20-iconv.ini, /etc/php/7.0/apache2/conf.d/20-json.ini, /etc/php/7.0/apache2/conf.d/20-mcrypt.ini, /etc/php/7.0/apache2/conf.d/20-mysqli.ini, /etc/php/7.0/apache2/conf.d/20-pdo_mysql.ini, /etc/php/7.0/apache2/conf.d/20-phar.ini, /etc/php/7.0/apache2/conf.d/20-posix.ini, /etc/php/7.0/apache2/conf.d/20-readline.ini, /etc/php/7.0/apache2/conf.d/20-shmop.ini, /etc/php/7.0/apache2/conf.d/20-sockets.ini, /etc/php/7.0/apache2/conf.d/20-sysvmsg.ini, /etc/php/7.0/apache2/conf.d/20-sysvsem.ini, /etc/php/7.0/apache2/conf.d/20-sysvshm.ini, /etc/php/7.0/apache2/conf.d/20-tokenizer.ini
PHP API	20151012
PHP Extension	20151012
Zend Extension	320151012
Zend Extension Build	API320151012,NTS
PHP Extension Build	API20151012,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	disabled
Zend Memory Manager	enabled
Zend Multibyte Support	disabled
IPv6 Support	enabled
DTrace Support	enabled
Registered PHP Streams	https, ftps, compress.zlib, php, file, glob, data, http, ftp, phar
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, tls, tlsv1.0, tlsv1.1, tlsv1.2
Registered Stream Filters	zlib.*, string.rot13, string.toupper, string.tolower, string.strip_tags, convert.*, consumed, dechunk, convert.iconv.*, mcrypt.*, mdecrypt.*

Esta página básicamente da información sobre tu servidor desde el punto de vista de PHP. Es útil para asegurar algunos ajustes o verificar que estás haciendo bien algunas cosas.

Si esto salió bien, significa que has hecho correctamente todos los pasos.
Si lo deseas, puedes eliminar este archivo (puede darle información sobre tu servidor a alguien que acceda a él).

Se elimina con:

```
sudo rm /var/www/html/info.php
```

Si lo eliminas, siempre puedes volver a crearlo.

Wordpress

WordPress es el CMS (Sistema de Gestión de Contenido) más popular en Internet. Te permite configurar sitios y blogs fácilmente sobre MySQL y PHP manejando el back-end. Wordpress ha tenido amplia aceptación y es una buena opción para tener un sitio en línea rápidamente. Después de la configuración, todo lo que queda hacer es partir a la administración a través del front-end.

En este tutorial, nos enfocaremos en crear una instalación Wordpress en una instancia LAMP (Linux, Apache, MySQL y PHP) sobre un servidor Ubuntu 18.04.

Paso 1: Crear una base de datos y usuario para Wordpress

El primer paso que vamos a tomar es el de preparación. Wordpress utilizar MySQL para manejar y almacenar la información del sitio y el usuario. Tenemos MySQL instalado, pero aún necesita una base de datos y su respectivo usuario para el uso de Wordpress.

Para iniciar, accede a la cuenta MySQL root (administrativa) utilizando el siguiente comando:

```
mysql -u root -p
```

Se te preguntará por la contraseña para el usuario root de MySQL que configuraste cuando instalaste el software.

Primero, podemos crear una base de datos separada que WordPress pueda controlar. Puedes llamarla como lo desees, en este caso utilizaremos “wordpress” para recordarlo. Puedes crear la base de datos para WordPress escribiendo:

```
CREATE DATABASE wordpress DEFAULT CHARACTER SET utf8 COLLATE
utf8_unicode_ci;
```

> **Nota:** Cada sentencia de MySQL debe terminar en punto-y-coma (;). Asegúrate de que esté presente si te surge algún error.

A continuación, debemos crear un usuario único en MySQL que será utilizado exclusivamente para operar con nuestra nueva base de datos. Creando bases de datos y cuentas de un uso único es una buena idea desde la perspectiva de seguridad y administración. Utilizaremos el nombre “wordpressuser” en este caso. Puedes utilizar cualquiera que desees al igual que con la base de datos.

Crearemos esta cuenta, configuraremos una contraseña, y le daremos acceso a la base de datos creada previamente. Podemos hacer esto escribiendo los siguientes comandos. Recuerda utilizar una contraseña segura para su usuario de base datos:

```
GRANT ALL ON wordpress.* TO 'wordpressuser'@'localhost' IDENTIFIED BY
'password';
```

Ahora tienes una base de datos y una cuenta de usuario, cada uno hecho especialmente para WordPress. Debemos concluir este proceso refrescando los privilegios para que la instancia actual de MySQL reconozca los cambios realizados:

```
FLUSH PRIVILEGES;
```

Sal de MySQL escribiendo:

```
EXIT;
```

Paso 2: Instalar extensiones adicionales para PHP

Cuando configuramos nuestra instancia LAMP, solo necesitábamos un conjunto mínimo de extensiones para que PHP se comunicara con MySQL. WordPress y muchos de sus plugins requieren extensiones de PHP adicionales.

Ahora podemos descargar e instalar algunas de las extensiones de PHP más populares para utilizar con WordPress escribiendo:

```
sudo apt-get update
sudo apt-get install php-curl php-gd php-mbstring php-xml php-xmlrpc
```

> **Nota**: Cada plugin de WordPress cuenta con su propio conjunto de requerimientos. Algunos requieren que algunos paquetes adicionales de PHP sean instalados. Puedes revisar la documentación de sus plugins para conocer más al respecto. Si están disponibles, pueden ser instalados mediante `apt-get` tal y como lo hemos hecho ahora.

Ahora reiniciaremos Apache para que reconozca las nuevas extensiones en la próxima sección. Puedes reiniciar Apache escribiendo:

```
sudo system apache2 restart
```

Paso 3: Ajustar la configuración de Apache para permitir sobre-escritura y re-escritura para .htaccess

Posteriormente, crearemos ajustes menores para nuestra configuración de Apache. Actualmente, el uso de los archivos .htaccess está deshabilitado. WordPress y muchos de sus plugins utilizan estos archivos exclusivamente para ediciones dentro del directorio para comunicarse con el servidor web.

Adicionalmente, habilitaremos mod_rewrite, el cual es necesario para que los enlaces permanentes de WordPress funcionen correctamente.

Habilitar sobre-escritura por .htaccess

¿Qué es el archivo .htaccess?
El archivo .htaccess (hypertext Access) es un texto en formato ASCII que permite definir los parámetros de un directorio en una página web sin tener que editar el archivo de configuración del servidor completo.

Normalmente, los archivos .htaccess funcionan para poner restricciones de seguridad a un sitio web ya que desde él, se pueden bloquear IPs desconocidas, bots y también modificar las direcciones para hacerlas más amigables y/o seguras.

Abre el archivo de configuración primaria de Apache para hacer nuestro primer cambio:

```
sudo nano /etc/apache2/apache2.conf
```

Para permitir archivos .htaccess, necesitamos configurar la directiva AllowOverride dentro del bloque Directory apuntando a nuestro documento raíz. Para ello en la parte inferior del archivo, agregaremos el siguiente bloque:

```
<Directory /var/www/html/>
    AllowOverride All
</Directory>
```

Al finalizar, guarde y cierre el archivo.

Lo siguiente, será habilitar mod_rewrite para poder utilizar la función de enlaces permanentes de WordPress:

```
sudo a2enmod rewrite
```

Antes de implementar los cambios que hemos realizado, revisemos para asegurarnos de que no hemos cometido algún error de sintaxis:

```
sudo apache2ctl configtest
```

La respuesta debe ser un mensaje similar a este:

A la salida:

```
AH00558: apache2: Could not reliably determine the server's fully
qualified domain name, using 127.0.1.1. Set the 'ServerName' directive
globally to suppress this message
Syntax OK
```

Si deseas reprimir la línea de arriba, solo agrega la directiva ServerName al archivo `/etc/apache2/apache2.conf` apuntando al dominio o IP de su servidor. En todo caso, este es solo un mensaje y no afectará la funcionalidad del sitio. Mientras que la respuesta contenga `Syntax OK`, estamos listos para continuar.

Reiniciaremos Apache para implementar los cambios:

```
sudo system apache2 restart
```

Paso 4: Descargar WordPress

Ahora que nuestro software está configurado en el servidor, podemos descargar y configurar WordPress. Por seguridad, siempre es recomendable obtener la versión más reciente de WordPress desde el sitio oficial.

Muévete a un directorio con permiso de escritura y posteriormente descargue la versión comprimida escribiendo:

```
cd /tmp
curl -O https://wordpress.org/latest.tar.gz
```

Extraiga el archivo comprimido para crear la estructura de directorios de WordPress:

```
tar xzvf latest.tar.gz
```

Moveremos esos archivos a nuestro documento raíz en su momento. Antes de hacerlo, podemos agregar un archivo .htaccess de prueba y configurar sus permisos, de manera que esté disponible para ser utilizado por WordPress posteriormente.

Cree el archivo y configure los permisos escribiendo:

```
touch /tmp/wordpress/.htaccess
chmod 660 /tmp/wordpress/.htaccess
```

Además, copiaremos el archivo de configuración ejemplo a un archivo que WordPress pueda leer:

```
cp /tmp/wordpress/wp-config-sample.php /tmp/wordpress/wp-config.php
```

Además, crearemos el directorio upgrade, para que WordPress no tenga conflictos posteriormente cuando intente hacerlo por sí mismo al actualizar el software.

```
mkdir /tmp/wordpress/wp-content/upgrade
```

Ahora podemos copiar todo el contenido del directorio en nuestro documento raíz. Utilizaremos `-a` para asegurarnos de que nuestros permisos permanezcan. Utilizaremos un punto al final de nuestro directorio fuente para indicar que todo lo que está dentro debe ser copiado, incluyendo los archivos ocultos (como el archivo .htaccess que creamos):

```
sudo cp -a /tmp/wordpress/. /var/www/html
```

Paso 5: Configurar el directorio WordPress

Antes de continuar con el sistema de configuración automático de WordPress, necesitamos ajustar algunos detalles en nuestro directorio.

Ajustando permisos y autoridad

Uno de los grandes detalles que necesitamos prevenir, es configurar apropiadamente los permisos y autoridad. Necesitamos ser capaces de escribir en esos archivos con un usuario regular, y necesitamos que el servidor web también sea capaz de ajustar algunos archivos y directorios para un funcionamiento correcto.

Empezaremos asignando la propiedad de todos los archivos en nuestro documento raíz a nuestro usuario. Utilizaremos `paula` como nuestro usuario en esta guía, pero deberías cambiar a uno que funcione con sudo en tu servidor. Asignaremos el grupo de propiedad al grupo www-data:

```
sudo chown -R paula:www-data /var/www/html
```

Posteriormente, configuraremos setgid a cada uno de nuestros directorios en nuestro directorio raíz. Esto causará que todos los archivos nuevos creados en esos directorios tengan el grupo del directorio padre (el cual debe ser www-data) en lugar del grupo del usuario primario. Esto solo para asegurarnos de que cuando sea que creemos un archivo en ese directorio mediante la línea de comandos, el servidor web seguirá siendo propietario del mismo.

Podemos configurar el bit setgid en todos los directorios de nuestra instalación de WordPress escribiendo:

```
sudo find /var/www/html -type d -exec chmod g+s {} \;
```

Hay algunos permisos más que debemos ajustar. Primero, le daremos al grupo acceso de escritura a la carpeta wp-content para que la interfaz web pueda realizar cambios en nuestros temas y plugins:

```
sudo chmod g+w /var/www/html/wp-content
```

Como parte de este proceso, le daremos al servidor web acceso a nuestro contenido en estos dos directorios:

```
sudo chmod -R g+w /var/www/html/wp-content/themes
sudo chmod -R g+w /var/www/html/wp-content/plugins
```

Estos permisos deberían ser suficientes para empezar. Algunos plugins y procedimientos pueden requerir ajustes adicionales.

Configurando el archivo de configuración de WordPress

Ahora, necesitamos hacer algunos cambios adicionales al archivo de configuración de WordPress.

Tan pronto abrimos el archivo, nuestra primera orden será ajustar algunas llaves privadas para proporcionar seguridad a nuestra instalación. WordPress proporciona un generador seguro para estos valores por lo que no deberá preocuparse de generar valores usted mismo Estos serán utilizados internamente, así que no hace daño que sean valores complejos.

Para obtener valores seguros desde el generador de WordPress, escriba:

```
curl -s https://api.wordpress.org/secret-key/1.1/salt/
```

Obtendrá un valor único que debe verse así:

> **¡¡¡Atención!!!** Es importante que solicites valores únicos cada vez. ¡NO copies los valores que se muestran abajo!

A la salida:

```
define('AUTH_KEY',         '1jl/vqfs<X**********************+c9.k<J@4H');
define('SECURE_AUTH_KEY',  'E2N-h2]Dcv**********************xf})CgLi-3');
define('LOGGED_IN_KEY',    'W(50,{W^,O**********************R6DUth[;88');
define('NONCE_KEY',        'll,4UC)7ua**********************zM7 o^-C7g');
define('AUTH_SALT',        'koMrurzOA+**********************D&?3w!BT#-');
define('SECURE_AUTH_SALT', 'p32*p,]z%L**********************+F|0h{!_xY');
define('LOGGED_IN_SALT',   'i^/G2W7!-1**********************[Hi])-qS`|');
define('NONCE_SALT',       'Q6]U:K?j4L**********************n+6&xqHN&%');
```

Estas son líneas de configuración que deberemos pegar directamente en nuestro archivo de configuración para crear llaves seguras. Copia la salida recibida ahora.

Ahora, abre el archivo de configuración:

```
nano /var/www/html/wp-config.php
```

Encuentra la sección que contiene valores simples para esos ajustes. Eso debe verse algo así:

```
define('AUTH_KEY',         'put your unique phrase here');
define('SECURE_AUTH_KEY',  'put your unique phrase here');
define('LOGGED_IN_KEY',    'put your unique phrase here');
define('NONCE_KEY',        'put your unique phrase here');
define('AUTH_SALT',        'put your unique phrase here');
define('SECURE_AUTH_SALT', 'put your unique phrase here');
define('LOGGED_IN_SALT',   'put your unique phrase here');
define('NONCE_SALT',       'put your unique phrase here');
```

Borra esas líneas y pegue las que ha copiado de la línea de comandos:

```
define('AUTH_KEY',         'VALORES COPIADOS DE LA LÍNEA DE COMANDOS');
define('SECURE_AUTH_KEY',  'VALORES COPIADOS DE LA LÍNEA DE COMANDOS');
define('LOGGED_IN_KEY',    'VALORES COPIADOS DE LA LÍNEA DE COMANDOS');
define('NONCE_KEY',        'VALORES COPIADOS DE LA LÍNEA DE COMANDOS');
define('AUTH_SALT',        'VALORES COPIADOS DE LA LÍNEA DE COMANDOS');
define('SECURE_AUTH_SALT', 'VALORES COPIADOS DE LA LÍNEA DE COMANDOS');
define('LOGGED_IN_SALT',   'VALORES COPIADOS DE LA LÍNEA DE COMANDOS');
define('NONCE_SALT',       'VALORES COPIADOS DE LA LÍNEA DE COMANDOS');
```

Lo siguiente, será modificar algunos ajustes de conexión de base de datos al inicio del archivo. Necesitará ajustar el nombre de base de datos, el usuario y la contraseña asociada que configuramos previamente en MySQL.

El otro cambio que debemos hacer, es configurar el método que WordPress deberá utilizar para escribir en el sistema de archivos. Debido a que ya hemos dado al servidor web los permisos necesarios, podemos explícitamente configurar el directorio del sistema a "direct". Una mala configuración con nuestros ajustes actuales, puede resultar en que WordPress solicite credenciales FTP cuando realicemos determinadas acciones.

Este ajuste puede ser agregar bajo las configuraciones de base de datos, o en cualquier otro lugar del archivo.

```
define('DB_NAME', 'wordpress');

/** MySQL database username */
define('DB_USER', 'wordpressuser');

/** MySQL database password */
define('DB_PASSWORD', 'password');

. . .

define('FS_METHOD', 'direct');
```

Guarda y cierra el archivo cuando termines.

Paso 6: Completar la instalación a través de la interfaz web

Ahora que la configuración del servidor está completa, podemos concluir el proceso de instalación desde la interfaz web.

En tu navegador, ve al dominio o la dirección IP pública de su servidor:

```
http://dominio_o_IP
```

Seleccione el idioma que desee utilizar.
Lo siguiente, es ir a la página inicial de configuración.

Seleccione el nombre de su sitio WordPress y escoja un nombre de usuario (se recomienda no utilizar algo como "admin" por seguridad). Una contraseña completa es generada automáticamente. Guarde esta contraseña o seleccione una contraseña compleja alternativa.

Introduzca su dirección de correo electrónico y seleccione si desea o no disuadir a los motores de búsqueda de indexar su sitio. Cuando haga clic, ahora será enviado a una página que le solicitará identificarse. Una vez que se haya identificado, será enviado al panel de administración de WordPress.

Actualizando WordPress

Cuando haya actualizaciones de WordPress, no podrás instalarlas mediante la interfaz web con los permisos actuales.

Los permisos que seleccionamos fueron pensados para obtener un balance entre seguridad y usabilidad para el 99% de las veces entre actualizaciones. De igual manera, hay un bit para restringir al software para aplicar actualizaciones automáticas.

Cuando una actualización esté disponible, inicia sesión nuevamente en su servidor con un usuario sudo. Temporalmente proporciona al proceso del servidor web acceso para el documento raíz completamente.

```
sudo chown -R www-data /var/www/html
```

Ahora, vuelve al panel de administración de WordPress y realiza las actualizaciones.

Cuando haya finalizado, devuelve los permisos a su estado anterior por seguridad:

```
sudo chown -R paula /var/www/html
```

Esto será únicamente necesario cuando realices actualizaciones del propio WordPress.

PHP y HTML

A continuación, voy a hablar sobre la programación en PHP y HTML. Esto serán un par de archivos que se encontrarán en nuestro servidor creado anteriormente con LAMP (Wordpress no hace falta). Se guardarán en el directorio `/var/www/html/`. En mi caso, he creado un directorio llamado "login" y trabajaré en él. Dentro de este, tengo dos archivos: `login.php` y `process.php`. `login.php` es el principal que llama a `process.php`.
Para ver el funcionamiento tenemos que escribir en el navegador:
`http://www.la_ip_del_servidor/login(en mi caso)/login.php`

En mi caso, no escribo directamente `ip_del_servidor/login.php` porque el archivo no está localizado en el directorio `/var/www/html/` sino que se encuentra además en un directorio dentro de este llamado login.

Primer código

Voy a comentar un código que lo que hace es crear en HTML lo siguiente:

Eso se hace con un sencillo código:

```
<!DOCTYPE html>
<html>
<head>
	<title>Mi pagina</title>
</head>
<body>
	<div id="frm">
		<form action="process.php" method="POST">
			<p>
				<label>Username:</label>
				<input type="text" id="user" name="user"/>
			</p>
			<p>
				<label>Password:</label>
				<input type="password" id="pass" name="pass"/>
			</p>
			<p>
				<input type="submit" id="btn" value="Login"/>
			</p>
		</form>
	</div>
</body>
</html>
```

Paso a paso

Esto indica que va a ser un código en HTML.

```
<!DOCTYPE html>
```

Se abre la parte de HTML y escribimos en la cabecera (head), es decir, el nombre que va a tener la pestaña o ventana cuando abramos este HTML en el navegador.

```
<html>
<head>
      <title>Mi pagina</title>
</head>
```

Tras la cabecera ya viene el cuerpo, lo que tiene el contenido (body), también decimos que esta parte del código se va a llamar POST y que este código va a llamar al archivo `process.php` (el siguiente código).

```
<body>
      <div id="frm">
            <form action="process.php" method="POST">
```

El primer objeto que creamos es una etiqueta en la que hay escrito "`Username:`" y a su lado un campo vacío de tipo texto llamado "user" en el que se escribirá el nombre de usuario.

```
<p>
      <label>Username:</label>
      <input type="text" id="user" name="user"/>
</p>
```

El segundo objeto que creamos es una etiqueta en la que hay escrito "`Password:`" y a su lado un campo vacío de tipo password (saldrán asteriscos o puntos en vez de los caracteres que se escriban) llamado "pass" en el que se escribirá la contraseña.

```
<p>
     <label>Password:</label>
     <input type="password" id="pass" name="pass"/>
</p>
```

El tercer y último objeto que creamos es un botón en el que hay escrito "`Login`", es decir, Iniciar sesión.

```
<p>
     <input type="submit" id="btn" value="Login"/>
</p>
```

Esto último cierra todas las partes que se han ido abriendo.

```
          </form>
     </div>
</body>
</html>
```

Segundo código

El segundo código es un poco más complicado. Lo que hace es recoger los datos introducidos en el anterior código, se conecta a una base de datos en MySQL y si en la tabla mitabla existe ese usuario con esa contraseña, lo muestra en una tabla con su id (autoinc.) que se le asignó automáticamente al ingresar los campos del usuario y la contraseña en la tabla de la base de datos "prueba".

```
<?php
        $username = filter_input(INPUT_POST, 'user');
        $password = filter_input(INPUT_POST, 'pass');
        $host = "localhost";
        $dbusername = "user";
        $dbpassword = "userpassword";
        $dbname = "prueba";
        $conn = new mysqli ($host, $dbusername, $dbpassword, $dbname);
?>
<!DOCTYPE html>
<html>
<head>
        <title>Mi pagina</title>
</head>
<body>
<br>
        <table>
                <tr>
                        <th>id</th>
                        <th>username</th>
                        <th>password</th>
                </tr>
                <?php
                $sql = "select * from mitabla where username =
'$username' and password = '$password'";
                        $result = mysqli_query($conn, $sql);
                        while($mostrar=mysqli_fetch_array($result)){
                ?>
                <tr>
                        <td><?php echo $mostrar['id'] ?></td>
                        <td><?php echo $mostrar['username'] ?></td>
                        <td><?php echo $mostrar['password'] ?></td>
                </tr>
                <?php
                }
                ?>
        </table>
</body>
</html>
```

Paso a paso

Se abre el código en PHP.

```
<?php
```

Coge los valores de login.php.

```
$username = filter_input(INPUT_POST, 'user');
$password = filter_input(INPUT_POST, 'pass');
```

Conexión al server y elección de base de datos, primero por partes (el host, nombre del usuario de la base de datos, contraseña de la base de datos, y el nombre de la base de datos) para después hacer la conexión más sencilla a la vista.

```
$host = "localhost";
$dbusername = "user";
$dbpassword = "userpassword";
$dbname = "prueba";

$conn = new mysqli ($host, $dbusername, $dbpassword, $dbname);
```

Se cierra el código en PHP y abrimos código en HTML.

```
?>

<!DOCTYPE html>
```

Al igual que en el código anterior, el nombre que va a tener la cabecera va a ser “Mi pagina”

```
<html>
<head>
        <title>Mi pagina</title>
</head>
```

En el cuerpo vamos a crear una tabla para mostrar a la salida el id, el nombre de usuario y la contraseña en negrita las tres (th en vez de td).

```
<body>
<br>
        <table>
                <tr>
                        <th>id</th>
                        <th>username</th>
                        <th>password</th>
                </tr>
```

Dentro del código en HTML abrimos un código en PHP y vamos a hacer una consulta en SQL en la conexión a la tabla y base de datos que realizamos anteriormente, la consulta lo que hace es buscar en la tabla “mitabla” (una tabla anteriormente creada) los registros que coincidan con los introducidos.

```
                <?php

                $sql = "select * from mitabla where username =
'$username' and password = '$password'";

                        $result = mysqli_query($conn, $sql);
```

Creamos un bucle while en el que mientras que la variable mostrar en la que está el vector con el resultado de la consulta tenga datos dentro, va a mostrar los datos que coinciden con las tres columnas id, username y password que se crearon antes.

```
                while($mostrar=mysqli_fetch_array($result)){
        ?>
        <tr>
                <td><?php echo $mostrar['id'] ?></td>
                <td><?php echo $mostrar['username'] ?></td>
                <td><?php echo $mostrar['password'] ?></td>
        </tr>
```

Abrimos un código PHP cortito para cerrar el bucle while y cerramos la tabla, el cuerpo y el código HTML.

```
        <?php
        }
        ?>
    </table>
</body>
</html>
```

Por ejemplo, al introducir en Username “testuser” y en Password “testuserpass” nos devuelve lo siguiente:

id	username	password
1	testuser	testuserpass

Es decir, que en la tabla en MySQL que creamos (mitabla2) hay un registro en el que el id = 1, username = testuser y password = testuserpass.