
Práctica 1.9

Implantación de Aplicaciones Web

Índice general

1	Arquitectura de una aplicación web LAMP en dos niveles	1
1.1	Arquitectura de dos niveles	2
1.2	Configuración de MySQL	2
1.3	Asignando privilegios a los usuarios de MySQL con GRANT	3
1.4	Comprobamos la lista de usuarios de MySQL	5
1.5	Comprobamos que podemos conectarnos a MySQL	6
1.6	Configuración de la aplicación web para conectar a MySQL	7
1.7	¿Qué puedo hacer si desde la máquina de <i>front-end</i> (Apache HTTP Server) no puedo conectar a la máquina de <i>back-end</i> (MySQL)?	7
1.8	Entregables	8
1.8.1	Documento técnico	8
1.8.2	Scripts de Bash	9
2	Referencias	10
3	Licencia	11

1 Arquitectura de una aplicación web LAMP en dos niveles

En esta práctica deberá automatizar la instalación y configuración de una aplicación web [LAMP](#) en **dos máquinas virtuales EC2** de [Amazon Web Services \(AWS\)](#), con la última versión de [Ubuntu Server](#). En una de las máquinas deberá instalar [Apache HTTP Server](#) y los módulos necesarios de [PHP](#) y en la otra máquina deberá instalar [MySQL Server](#).

Ahora vamos a tener la [pila LAMP](#) repartida en dos máquinas virtuales, una se encargará de gestionar las peticiones web y la otra de gestionar la base de datos.

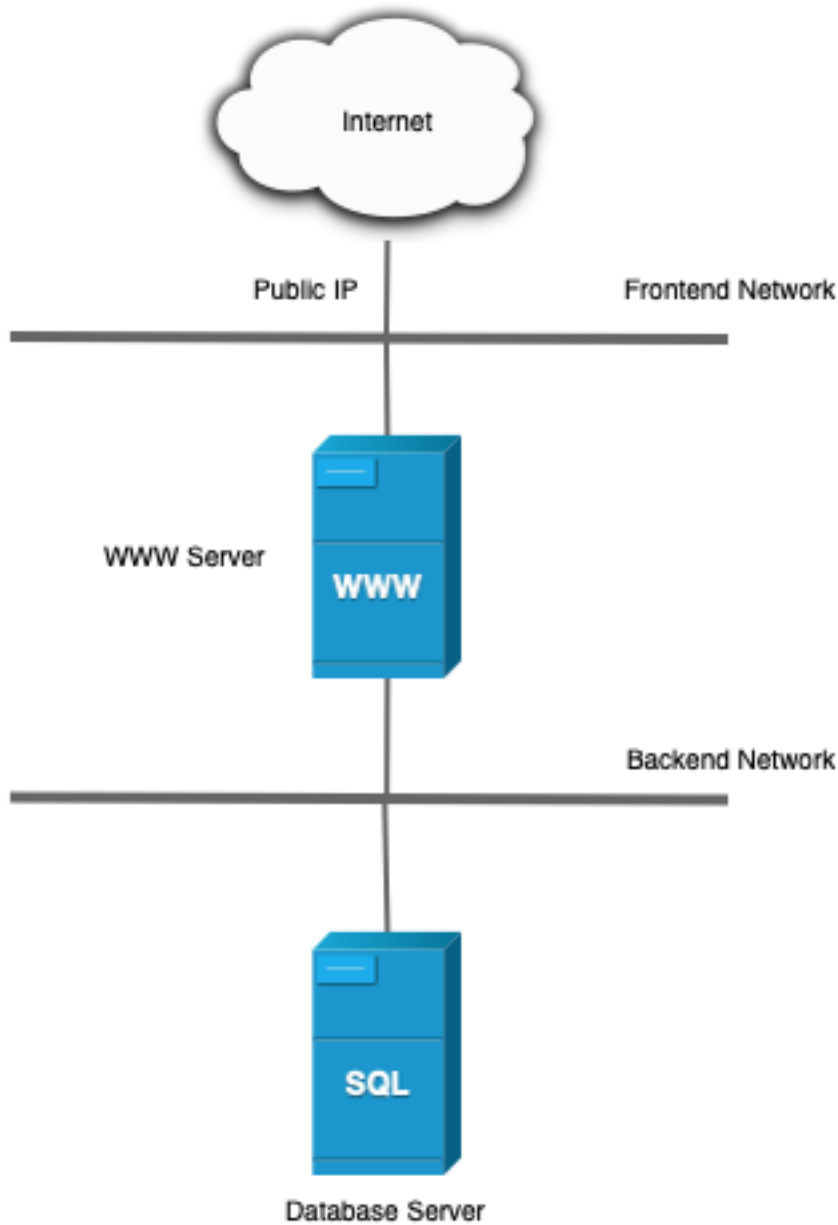
Una vez que hayas comprobado que todos los servicios de la [pila LAMP](#) están funcionando correctamente en las dos máquinas, instala y configura la aplicación propuesta.

Ten en cuenta que tendrás que modificar la configuración de [MySQL Server](#) para que permita conexiones remotas y también tendrás que revisar los privilegios del usuario que se conecta a la base de datos de la aplicación.

La arquitectura estará formada por:

- Una capa de *front-end*, formada por un servidor web con [Apache HTTP Server](#).
- Una capa de *back-end*, formada por un servidor [MySQL](#).

1.1. Arquitectura de dos niveles



1.2. Configuración de MySQL

Edita el siguiente archivo de configuración:

```
1 /etc/mysql/mysql.conf.d/mysqld.cnf
```

Busca la directiva de configuración `bind-address` dentro del bloque de `[mysqld]`:

```
1 [mysqld]
2 bind-address = 127.0.0.1
```

En la configuración por defecto, MySQL sólo permite conexiones desde localhost (127.0.0.1). Habrá que modificar este valor por la dirección IP de la máquina donde se está ejecutando el servicio de MySQL.

```
1 [mysqld]
2 bind-address = IP_SERVIDOR_MYSQL
```

El valor de `IP_SERVIDOR_MYSQL` indica desde que **interfaz de red del servidor de MySQL** se van a permitir conexiones. En nuestro caso, los valores que podemos poner aquí son:

- **IP pública de AWS:** Permitimos el acceso a cualquier máquina que envíe una petición de conexión desde la red pública de AWS.
- **IP privada de AWS:** Sólo permitimos el acceso desde máquinas que estén en la misma red privada de AWS.
- **127.0.0.1:** No permitimos conexiones remotas, sólo se permiten conexiones dentro de la misma máquina.
- **0.0.0.0:** Indica que permitimos conexiones desde cualquier interfaz de red que tenga la máquina.

Si nuestra máquina dispone más de una interfaz de red podemos poner la dirección IP `0.0.0.0` para permitir que se puedan conectar a MySQL desde cualquiera de las interfaces de red disponibles.

```
1 [mysqld]
2 bind-address = 0.0.0.0
```

Una vez hecho esto tenemos que reiniciar el servicio de MySQL:

```
1 sudo systemctl restart mysql
```

1.3. Asignando privilegios a los usuarios de MySQL con GRANT

Si queremos mejorar la seguridad de nuestro servidor de MySQL, podemos configurar que el usuario de MySQL sólo pueda conectarse desde una determinada dirección IP.

Paso 1. Eliminar si existe algún usuario previo con el mismo nombre

Antes de crear un usuario, debemos comprobar si existe y eliminarlo en caso de que exista. Para ello, ejecutamos la siguiente sentencia SQL:

```
1 DROP USER IF EXISTS '$DB_USER'@$IP_MAQUINA_CLIENTE';
```

Donde `$DB_USER` es el nombre del usuario de MySQL y `$IP_MAQUINA_CLIENTE` es la dirección IP desde la que el usuario puede conectarse al servidor de MySQL.

Paso 2. Creación de un usuario

La sintaxis para crear un usuario de MySQL es la siguiente:

```
1 CREATE USER '$DB_USER'@$IP_MAQUINA_CLIENTE' IDENTIFIED BY '$DB_PASS';
```

Donde `$DB_USER` es el nombre del usuario de MySQL, `$DB_PASS` es la contraseña y `$IP_MAQUINA_CLIENTE` es la dirección IP desde la que el usuario podrá conectarse al servidor de MySQL.

Los valores que podemos utilizar para `$IP_MAQUINA_CLIENTE` son los siguientes:

- **localhost**: No permitimos conexiones remotas, sólo se permiten conexiones dentro de la misma máquina.
- **%**: Este comodín indica que permitimos conexiones desde cualquier máquina.
- **Una dirección IP**: Para que el usuario solo pueda conectarse al servidor de MySQL desde esa dirección. Ejemplo: 172.31.10.11.
- **Una dirección IP con un comodín**: Para permitir que el usuario pueda conectarse desde rango de direcciones IPs. Ejemplo: 172.31.%.

Tenga en cuenta que tendrá que reemplazar los valores de las variables `$DB_USER`, `$DB_PASS` y `$IP_MAQUINA_CLIENTE` por los valores que necesite.

Paso 3. Asignación de privilegios

Una vez que hemos creado el usuario le asignamos los privilegios que sean necesarios sobre la base de datos de la aplicación. Para asignar privilegios utilizaremos la sentencia `GRANT` de SQL. La **sintaxis** es la siguiente:

```
1 GRANT [permiso] ON [nombre_base_de_datos].[nombre_tabla] TO '$DB_USER'@$IP_MAQUINA_CLIENTE';
```

Por ejemplo:

```
1 GRANT ALL PRIVILEGES ON *.* TO 'nombre_usuario'@'localhost';
```

En este comando, los asteriscos indican que estamos aplicando el permiso `ALL PRIVILEGES` al usuario `nombre_usuario` para que pueda acceder a todas las tablas de cada una de las bases de datos. En el ejemplo anterior, el usuario sólo puede conectarse al servidor de MySQL desde `localhost`.

En esta práctica tenemos que asignar todos los privilegios con `ALL PRIVILEGES` al usuario de MySQL que vamos a utilizar para conectarnos desde la máquina donde está corriendo el servicio de Apache HTTP.

```
1 GRANT ALL PRIVILEGES ON '$DB_NAME'.* TO '$DB_USER'@$IP_MAQUINA_CLIENTE';
```

Tenga en cuenta que tendrá que reemplazar los valores de las variables `$DB_NAME`, `$DB_USER` y `$IP_MAQUINA_CLIENTE` por los valores que necesite.

Los diferentes tipos de permisos que podemos aplicar son los siguientes:

- **ALL PRIVILEGES**: permite a un usuario de MySQL acceder a todas las bases de datos asignadas en el sistema.
- **CREATE**: permite crear nuevas tablas o bases de datos.
- **DROP**: permite eliminar tablas o bases de datos.
- **DELETE**: permite eliminar registros de tablas.
- **INSERT**: permite insertar registros en tablas.

- **SELECT**: permite leer registros en las tablas.
- **UPDATE**: permite actualizar registros seleccionados en tablas.
- **GRANT OPTION**: permite asignar privilegios a otros usuarios.

Paso 4. Actualizamos las tablas de privilegios

Para que los cambios que hemos realizado sobre los privilegios de los usuarios se apliquen de forma inmediata, tendremos que ejecutar el siguiente comando.

```
1 FLUSH PRIVILEGES;
```

Ejemplo

En el script de bash de la práctica que va a realizar debería tener el siguiente bloque de instrucciones SQL para crear el usuario de MySQL y asignarle los privilegios necesarios sobre la base de datos de la aplicación web:

```
1 DROP USER IF EXISTS '$DB_USER'@$IP_MAQUINA_CLIENTE';
2 CREATE USER '$DB_USER'@$IP_MAQUINA_CLIENTE IDENTIFIED BY '$DB_PASS';
3 GRANT ALL PRIVILEGES ON '$DB_NAME'.* TO '$DB_USER'@$IP_MAQUINA_CLIENTE';
```

Las variables estarán definidas en un archivo `.env`:

1.4. Comprobamos la lista de usuarios de MySQL

Los usuarios de MySQL se almacenan en la tabla `mysql.user`. La clave primaria de esta tabla está formada por los valores `user` y `host`, de modo que cada fila vendrá identificada por un nombre de usuario y el host desde el que puede conectarse.

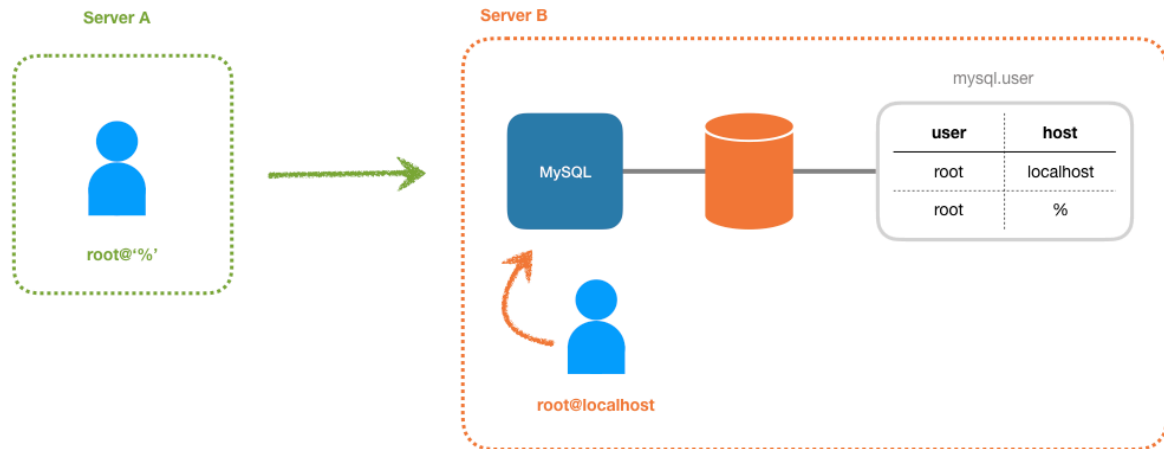
La siguiente consulta nos devuelve el listado de usuarios que tenemos en MySQL y desde qué host pueden conectarse:

```
1 SELECT user,host FROM mysql.user;
```

En nuestra caso la consulta anterior devuelve el siguiente resultado:

```
1 +-----+-----+
2 | user          | host          |
3 +-----+-----+
4 | root          | %             |
5 | root          | localhost     |
6 | lamp_user     | %             |
7 | lamp_user     | localhost     |
8 | debian-sys-maint | localhost     |
9 | phpmyadmin    | localhost     |
10 | mysql.session | localhost     |
11 | mysql.sys     | localhost     |
12 +-----+-----+
```

El siguiente diagrama muestra un ejemplo de dos usuarios que se están conectando a una máquina con MySQL Server. El usuario `root@localhost` es un usuario que sólo puede conectarse desde la máquina local y el usuario `root@' %'` es un usuario que se puede conectar desde una máquina remota.



También podemos consultar qué permisos específicos tiene un determinado usuario. La siguiente consulta nos devuelve los permisos que tiene el usuario `root`:

```
1 SHOW GRANTS FOR root;
```

```
1 +-----+
2 | Grants for root@% |
3 +-----+
4 | GRANT ALL PRIVILEGES ON *.* TO 'root'@'%' |
5 +-----+
```

1.5. Comprobamos que podemos conectarnos a MySQL

Ahora vamos a comprobar que podemos conectarnos con MySQL desde la máquina donde está corriendo el servicio de Apache HTTP. Podemos comprobarlo conectando con el shell de `mysql`:

```
1 mysql -u USERNAME -p -h IP-SERVIDOR-MYSQL
```

O haciendo un telnet al puerto donde está corriendo el servicio de MySQL:

```
1 telnet IP-SERVIDOR-MYSQL 3306
```

Si no podemos conectarnos a MySQL revisaremos que el servicio está activo y que no tenemos ningún firewall que nos esté filtrando el puerto del servicio donde se ejecuta MySQL.

1.6. Configuración de la aplicación web para conectar a MySQL

Una vez que has realizado los pasos anteriores debes configurar la aplicación web para que pueda conectar a MySQL.

El repositorio de la aplicación web que tendrás que poner en producción es el siguiente:

- <https://github.com/josejuansanchez/iaw-practica-lamp>

En esta aplicación web el archivo de configuración que tendrás que editar y configurar se llama `config.php`.

El contenido del archivo `config.php` es el siguiente:

```
1 <?php
2
3 define('DB_HOST', 'localhost');
4 define('DB_NAME', 'database_name_here');
5 define('DB_USER', 'username_here');
6 define('DB_PASSWORD', 'password_here');
7
8 $mysqli = mysqli_connect(DB_HOST, DB_USER, DB_PASSWORD, DB_NAME);
9
10 ?>
```

En este archivo se definen las constantes `DB_HOST`, `DB_NAME`, `DB_USER` y `DB_PASSWORD`. Por lo tanto, estos son los valores que tendrás que configurar para que tu aplicación web pueda conectarse a MySQL.

1.7. ¿Qué puedo hacer si desde la máquina de *front-end* (Apache HTTP Server) no puedo conectar a la máquina de *back-end* (MySQL)?

A continuación se describe una secuencia de pasos que se pueden realizar para detectar posibles errores cuando desde la máquina de *front-end* (Apache HTTP Server) no puedo conectar con la máquina de *back-end* (MySQL).

- Lo primero que deberíamos hacer sería comprobar que las dos instancias están en ejecución.
- Una vez que hemos comprobado que las dos instancias están en ejecución, deberíamos comprobar que están en la misma subred y que pueden verse entre sí haciendo uso de la utilidad `ping`.
- Si las máquinas están en la misma subred y pueden verse entre sí, debemos comprobar que el puerto 3306 de la máquina donde se ejecuta MySQL está abierto. Revisa la configuración del firewall para verificar que el puerto 3306 está abierto.
- Una vez que hemos verificado que el puerto 3306 está abierto en el firewall, podemos comprobar si el servicio de MySQL está en ejecución en ese puerto. Para comprobarlo podemos hacer un `telnet` al puerto 3306 desde la máquina de *front-end*, donde se ejecuta Apache HTTP Server. Ejemplo: `telnet IP-SERVIDOR-MYSQL 3306`.

- Si el puerto está abierto recibirás algún mensaje de respuesta al hacer un `telnet`. Si no recibes ninguna respuesta comprueba que el parámetro `bind-address` está bien configurado en el archivo de configuración de MySQL, para que pueda aceptar conexiones desde la subred donde está la máquina de *front-end* donde se ejecuta Apache HTTP Server.
- Si el puerto 3306 está abierto, podemos comprobar que el usuario de MySQL de la aplicación web puede conectarse desde la máquina de *front-end* donde se ejecuta Apache HTTP Server. Recuerda que el usuario que se conecta desde otra máquina tiene que crearse con el comodín `%` en el valor del host. En la máquina de Apache puedes instalar el cliente de MySQL con `apt install mysql-client` y comprobar que puedes conectarte a MySQL con el usuario de la aplicación web. Ejemplo: `mysql -u USERNAME -p -h IP-SERVIDOR-MYSQL`.
- Comprueba que has configurado correctamente el archivo de configuración de la aplicación web para que pueda conectarse a MySQL.

1.8. Entregables

Deberá crear un repositorio en [GitHub](#) con el nombre de la práctica y añadir al profesor como colaborador.

El repositorio debe tener el siguiente contenido:

- Un **documento técnico** con la descripción de todos los pasos que se han llevado a cabo.
- Los **scripts de Bash** que se han utilizado para automatizar la instalación y configuración de la pila LAMP, así como de la [aplicación web propuesta](#).

Además del contenido anterior puede ser necesario crear otros archivos de configuración. A continuación se muestra un ejemplo de cómo puede ser la estructura del repositorio:

```
1  .|—
2  README.md|—
3  conf|
4    └─ 000-default.conf ─┘
5  scripts|—
6    .env|—
7    install_lamp_frontend.sh|—
8    install_lamp_backend.sh|—
9    setup_letsencrypt_https.sh ─┘
10   deploy.sh
```

1.8.1. Documento técnico

El documento técnico `README.md` tiene que estar escrito en [Markdown](#) y debe incluir **como mínimo** los siguientes contenidos:

- Descripción del proceso de instalación de la instalación de [la aplicación web propuesta](#) utilizando una arquitectura de dos niveles.

1.8.2. Scripts de Bash

El directorio `scripts` debe incluir los siguientes archivos:

- `.env`: Este archivo contiene todas las variables de configuración que se utilizarán en los scripts de Bash.
- `install_lamp_frontend.sh`: Script de Bash con la automatización del proceso de instalación de la pila LAMP en la máquina de frontend.
- `install_lamp_backend.sh`: Script de Bash con la automatización del proceso de instalación de la pila LAMP en la máquina de backend.
- `setup_letsencrypt_https.sh`: Script de Bash con la automatización del proceso de solicitar un certificado SSL/TLS de Let's Encrypt y configurarlo en el servidor web Apache.
- `deploy.sh`: Script de Bash con la automatización del proceso de instalación de WordPress sobre el directorio raíz `/var/www/html`.

2 Referencias

- [Amazon Web Services](#)
- [Ubuntu Server](#)
- [LAMP Stack](#)
- [PHP](#)
- [Apache HTTP Server](#)
- [MySQL Server](#)
- [Move MySQL to a Separate Cloud Database Server](#)
- [Markdown](#)

3 Licencia

Esta página forma parte del curso Implantación de Aplicaciones Web de José Juan Sánchez Hernández y su contenido se distribuye bajo una licencia Creative Commons Reconocimiento-NoComercial-CompartirIgual 4.0 Internacional.