

Workshop 3

Crear un servidor Debian con Base de Datos y conectarlo con el WebServer del workshop II

Pasos a seguir

1. Encendemos la máquina virtual WebServer

Maquina Anfritiona

```
vagrant ssh
```

Maquina virtual

```
sudo apachectl -S
```

Esto confirmará que el servidor Apache está funcionando correctamente y te mostrará la configuración del servidor.

2. Creamos otra máquina virtual para la Base de Datos

1. Creamos una carpeta nueva dentro de nuestra carpeta donde están las máquinas virtuales y entramos a dicha carpeta ya creada.

```
mkdir database  
cd database
```

2. Creamos un nuevo Vagrantfile para la máquina virtual con **Debian**.

```
vagrant init debian/bookworm64
```

3. Abrimos el archivo Vagrantfile en [VS Code](#) y editamos la línea 35 para ponerle una IP **Diferente** a la máquina WebServer que tiene la IP "192.168.56.10". La nueva máquina tendrá la IP "192.168.56.11".

```
code Vagrantfile
```

4. Encendemos la **nueva** máquina virtual.

```
vagrant up
```

5. Ejemplo del archivo Vagrantfile.

```
1 # Create a forwarded port mapping which allows access to a specific port
2 # within the machine from a port on the host machine and only allow access
3 # via 127.0.0.1 to disable public access
4 # config.vm.network "forwarded_port", guest: 80, host: 8080, host_ip: "127.0.0.1"
5
6 # Create a private network, which allows host-only access to the machine
7 # using a specific IP.
8 config.vm.network "private_network", ip: "192.168.56.11"
9
10 # Create a public network, which generally matched to bridged network.
11 # Bridged networks make the machine appear as another physical device on
12 # your network.
13 # config.vm.network "public_network"
```

3. Cambiamos el nombre del usuario/máquina para no confundirlo con las otras máquinas virtuales.

1. Abrimos el archivo de la **nueva** máquina donde está el nombre de esta y le cambiamos el nombre por **"database"**.

```
sudo nano /etc/hosts
```

2. Hacemos que la máquina detecte el cambio y nos salimos de la máquina y volvemos a entrar.

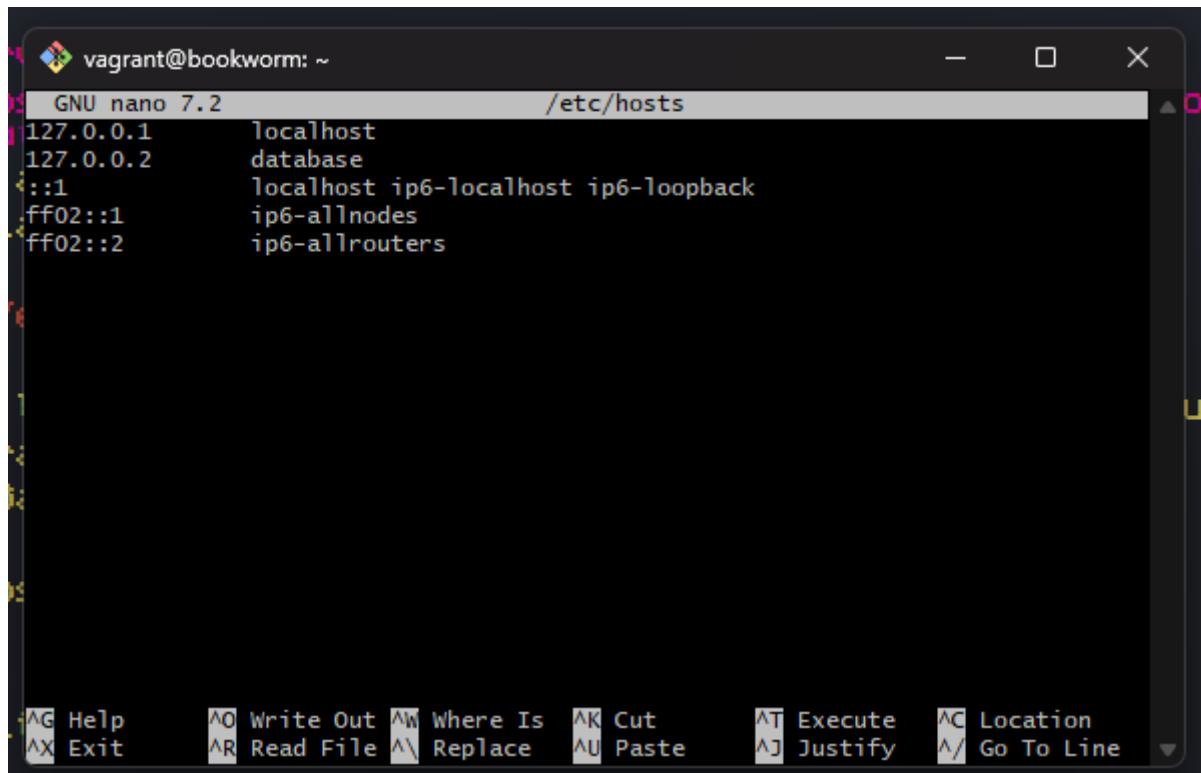
- Máquina DataBase

```
sudo hostnamectl set-hostname database
exit
```

- Máquina Anfritiona

```
vagrant ssh
```

3. Ejemplo de estructura de carpetas para cambiar el nombre:

A screenshot of a terminal window titled 'vagrant@bookworm: ~'. The terminal shows the GNU nano 7.2 editor editing the /etc/hosts file. The file content is as follows:

```
127.0.0.1    localhost
127.0.0.2    database
::1         localhost ip6-localhost ip6-loopback
ff02::1     ip6-allnodes
ff02::2     ip6-allrouters
```

The bottom of the terminal shows the nano editor's command palette with options like Help, Write Out, Where Is, Cut, Execute, Location, Exit, Read File, Replace, Paste, Justify, and Go To Line.

4. Realizamos el cambio del nombre de la misma forma con la máquina WebServer.

```
sudo nano /etc/hosts
```

- Máquina WebServer

```
sudo hostnamectl set-hostname webserver
exit
```

- Máquina Anfitrión

```
vagrant ssh
```

4. Instalamos los paquetes necesarios para configurar nuestra máquina y hacerla una Base de Datos.

1. Actualizamos los paquetes disponibles.

```
sudo apt-get update
```

2. Instalamos los paquetes necesarios para tener una base de datos MySQL (MariaDB en este caso).

```
sudo apt-get install mariadb-server mariadb-client  
mysql --version
```

3. Ejecutamos MySQL y empezamos a configurarlo.

```
sudo mysql
```

- En MySQL:

1. Vemos las bases de datos creadas.

```
show databases;
```

2. Vemos los usuarios creados y creamos uno que vamos a utilizar para que la máquina WebServer se conecte.

```
desc mysql.user;  
Select Host, User, Password from mysql.user;  
Create user laravel;  
  
Create user laravel2@'localhost';  
Drop user laravel2@'localhost';  
Drop user laravel;  
  
Create user laravel identified by 'secret';  
Set password for laravel = password('secret2');  
Set password for laravel = password('secret');
```

3. Creamos la base de datos que va a utilizar la máquina WebServer.

```
Create database lfts;
```

4. Le asignamos permisos al usuario que creamos anteriormente, refrescamos los privilegios y nos salimos de MySQL.

```
Grant all privileges on lfts.* to laravel;  
show grants for laravel;  
  
Flush privileges;  
  
exit
```

4. Probamos que funcione el usuario y vemos que bases de datos podemos utilizar.

```
mysql -u laravel -p
secret

Show databases;

exit
```

5. Accedemos desde la máquina WebServer a la base de datos.

1. Primero instalamos en la máquina WebServer los paquetes de MySQL, solo el cliente porque con este nos conectamos.

```
sudo apt-get install mariadb-client
mysql --version
```

2. Nos intentamos conectar a la máquina DataBase con el usuario que creamos.

```
mysql -h 192.168.56.11 -u laravel -p
secret
```

3. Vemos que no nos podemos conectar porque no hemos configurado la máquina DataBase para que permita las conexiones desde afuera.

```
sudo nano /etc/mysql/mariadb.conf.d/50-server.cnf
```

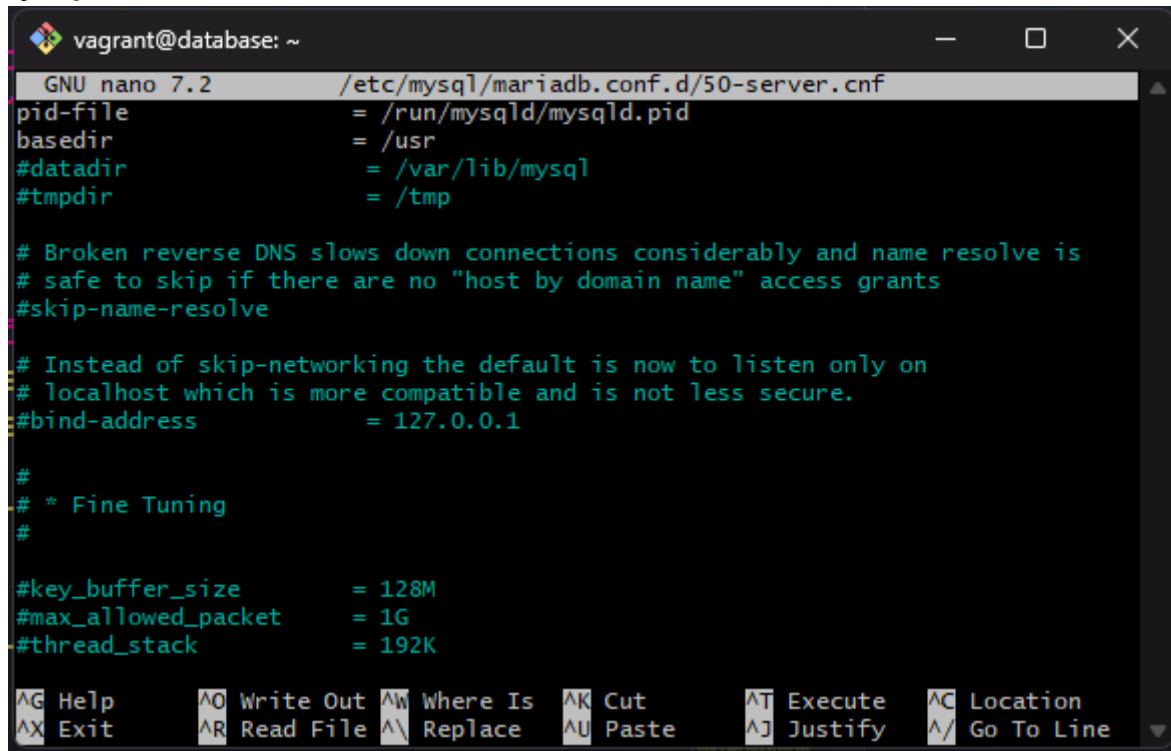
- Comentamos la línea:

```
#bind--adress          =127.0.0.1
```

- Reiniciamos el servicio MySQL.

```
sudo systemctl restart mysql
```

4. Ejemplo del archivo de servidor.



```

vagrant@database: ~
GNU nano 7.2 /etc/mysql/mariadb.conf.d/50-server.cnf
pid-file           = /run/mysqld/mysqld.pid
basedir            = /usr
#datadir           = /var/lib/mysql
#tmpdir            = /tmp

# Broken reverse DNS slows down connections considerably and name resolve is
# safe to skip if there are no "host by domain name" access grants
#skip-name-resolve

# Instead of skip-networking the default is now to listen only on
# localhost which is more compatible and is not less secure.
#bind-address      = 127.0.0.1

#
# * Fine Tuning
#

#key_buffer_size   = 128M
#max_allowed_packet = 1G
#thread_stack      = 192K

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute  ^C Location
^X Exit      ^R Read File ^_ Replace   ^U Paste     ^J Justify  ^/_ Go To Line
  
```

5. Volvemos a conectarnos desde la máquina WebServer ahora que modificamos el archivo de configuración para permitir la conexión externa.

```

mysql -h 192.168.56.11 -u laravel -p
secret

# In MySQL:
exit
  
```

6. Instalamos paquetes para poder hacer páginas con PHP usando Laravel.

1. Instalamos los paquetes necesarios para utilizar Laravel.

```

sudo apt-get install php8.2 php8.2-mysql php8.2-mcrypt php8.2-memcache
php8.2-bcmath php8.2-zip php8.2-curl php8.2-xml
  
```

2. Instalamos Composer para gestionar dependencias PHP.

```

php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');"
php -r "if (hash_file('sha384', 'composer-setup.php') ===
'dac665fdc30fdd8ec78b38b9800061b4150413ff2e3b6f88543c636f7cd84f6db9189d43a81
e5503cda447da73c7e5b6') { echo 'Installer verified'; } else { echo
'Installer corrupt'; unlink('composer-setup.php'); } echo PHP_EOL;"
php composer-setup.php
php -r "unlink('composer-setup.php');"
  
```

3. Creamos una carpeta para Composer y movemos algunos archivos a esa carpeta.

```
sudo mkdir -p /opt/composer
sudo mv composer.phar /opt/composer/
sudo ln -s /opt/composer/composer.phar /usr/local/bin
```

4. Creamos una acceso directo para el archivo composer.phar.

```
ls -la /usr/local/bin/c*
sudo ln -s /opt/composer/composer.phar /usr/local/bin/composer
sudo rm /usr/local/bin/composer.phar
```

5. Instalamos NVM (Node Version Manager) para gestionar versiones de Node.js.

- Instalamos curl

```
sudo apt-get install curl
curl ifconfig.me
```

- Instalamos NVM

```
curl -o- https://raw.githubusercontent.com/nvm-
sh/nvm/v0.39.7/install.sh | bash
```

- Salimos y volvemos a ingresar.

- Maquina WebServer

```
exit
```

- Maquina Anfritiona

```
vagrant ssh
```

- Verificamos la instalación.

```
nvm ls-remote
nvm install --lts
node -v
npm -v
```

6. Creamos el proyecto Laravel en la carpeta compartida entre la máquina anfitriona y la máquina virtual.

```
cd /vagrant/sites  
composer create-project laravel/laravel=8.6.12 lfts.isw811.xyz
```

7. Configuramos Apache para servir el proyecto Laravel.

1. Creamos una nueva configuración de VirtualHost para Apache.

```
sudo nano /etc/apache2/sites-available/lfts.isw811.xyz.conf
```

◦ Agregamos el siguiente contenido:

```
<VirtualHost *:80>  
    ServerAdmin webmaster@lfts.isw811.xyz  
    ServerName lfts.isw811.xyz  
  
    # Indexes + Directory Root.  
    DirectoryIndex index.php index.html  
    DocumentRoot /home/vagrant/sites/lfts.isw811.xyz/public  
  
    <Directory /home/vagrant/sites/lfts.isw811.xyz/public>  
        DirectoryIndex index.php index.html  
        AllowOverride All  
        Require all granted  
    </Directory>  
  
    ErrorLog ${APACHE_LOG_DIR}/lfts.isw811.xyz.error.log  
    LogLevel warn  
    CustomLog ${APACHE_LOG_DIR}/lfts.isw811.xyz.access.log combined  
</VirtualHost>
```


2. Ejemplo del archivo configuracion

```
1 <VirtualHost *:80>
2   ServerAdmin webmaster@primerapagina.com
3   ServerName primerapagina.com
4
5   # Indexes + Directory Root.
6   DirectoryIndex index.php index.html
7   DocumentRoot /home/vagrant/sites/primerapagina.com
8
9   <Directory /home/vagrant/sites/primerapagina.com>
10      DirectoryIndex index.php index.html
11      AllowOverride All
12      Require all granted
13   </Directory>
14
15   ErrorLog ${APACHE_LOG_DIR}/primerapagina.com.error.log
16   LogLevel warn
17   CustomLog ${APACHE_LOG_DIR}/primerapagina.com.access.log combined
18 </VirtualHost>
19
```

3. Activamos el sitio y recargamos Apache.

```
sudo cp lfts.isw811.xyz.conf /etc/apache2/sites-available/
sudo apache2ctl -t
sudo a2ensite lfts.isw811.xyz.conf
sudo apache2ctl -t
sudo systemctl reload apache2
```

4. Verificamos la instalación de Composer y ajustamos el archivo de configuración .env de Laravel para conectarse a la base de datos.

```
cd /vagrant/sites/lfts.isw811.xyz
nano .env
```

- Configuramos las siguientes variables:

```
DB_CONNECTION=mysql
DB_HOST=192.168.56.11
DB_PORT=3306
DB_DATABASE=lfts
DB_USERNAME=laravel
DB_PASSWORD=secret
```

5. Ejecutamos las migraciones de Laravel para crear las tablas en la base de datos.

```
composer --global config process-timeout 2000  
php artisan migrate
```

¡Listo! Ahora tienes una máquina Debian configurada como servidor de base de datos conectada con el WebServer usando Laravel. Asegúrate de probar tu sitio web en un navegador para verificar que todo funcione correctamente.

[My GitHub](#)