

Práctica UD2: Docker

Desenvolvemento de aplicacións web

MP0614. Despregamento de aplicacións web

Sumario

Instrucciones	3
1. Docker	4
1.1. Instalación de docker e imagen hello world.	4
1.2. Descarga imagen de ubuntu y accede a shell interactivo	4
1.3. Configura entorno de nodejs	4
1.4 Guarda cambios en la imagen de ubuntu y haz push a tu repositorio de docker.....	4
2. Docker compose.....	5
2.1 Instalación de Docker compose.....	5
2.2 Despliega app de prueba sobre servidor web nginx con fichero docker-compose.yml	5
2.3. Pila LAMP con Docker compose	5

Instrucciones

- Las capturas de las máquinas virtuales deben mostrar el nombre de la máquina.
- En el nombre de la máquina virtual debe contener la inicial y el apellido del alumno/a que entrega la práctica.
 - Por ejemplo, si creo una máquina virtual llamada "vsFTPd Server", debo nombrarla "jlopez vsFTPd Server".
- Las capturas deben de tener una calidad suficiente para que su contenido pueda ser legible.
- La entrega será en la tarea de la plataforma moodle mediante un fichero pdf practica_x_tu_nombre.pdf (x es número de practica y tu_nombre es tu nombre) en el que se puedan ver en las diferentes secciones lo solicitado.

1. Docker

1.1. Instalación de docker e imagen hello world.

Realiza la instalación de docker conforme se indica por ejemplo en este tutorial:

<https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-ubuntu-22-04>

Actualizamos el repositorio con:

apt-get update

Como prerequisite, necesitamos que apt pueda usar paquetes a través de HTTPS , para ello usamos el siguiente comando:

(si no lo tenemos debemos instalar curl con apt-get install curl)

sudo apt install apt-transport-https ca-certificates curl software-properties-common

```
root@debian12:/home/davidrl# sudo apt install apt-transport-https ca-certificates curl software-properties-common
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
apt-transport-https is already the newest version (2.6.1).
ca-certificates is already the newest version (20230311).
curl is already the newest version (7.88.1-10+deb12u7).
software-properties-common is already the newest version (0.99.30-4.1~deb12u1).
software-properties-common set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 1 not upgraded.
root@debian12:/home/davidrl#
```

Ahora añadimos la clave GPG del repositorio oficial de docker a nuestra maquina, para ello usamos el comando:

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg

```
root@debian12:/home/davidrl# curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
root@debian12:/home/davidrl#
```

Ahora añadimos los repositorios de docker a nuestro sources.list:

```
echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" |
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

```
root@debian12:/home/davidrl# echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.gpg]
https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
root@debian12:/home/davidrl#
```

Ahora actualizamos nuevamente los repositorios de apt:

apt-get update

Nos aseguramos de que vamos a instalar Docker desde su repositorio en lugar del repositorio por defecto de Debian:

apt-cache policy docker-ce

```
root@debian12:/home/davidrl# apt-cache policy docker-ce
docker-ce:
  Installed: (none)
  Candidate: 5:27.3.1-1~debian.12~bookworm
  Version table:
     5:27.3.1-1~debian.12~bookworm 500
        500 https://download.docker.com/linux/debian bookworm/stable amd64 Packages
     5:27.3.0-1~debian.12~bookworm 500
        500 https://download.docker.com/linux/debian bookworm/stable amd64 Packages
```

Ahora instalamos Docker:

apt install docker-ce

Una vez que Docker ha terminado de instalar, comprobamos que su servicio esta corriendo en la maquina:

service docker status

```
root@debian12:/home/davidrl# service docker status
• docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; preset: enabled)
   Active: active (running) since Sun 2024-10-13 13:32:47 CEST; 1min 35s ago
   TriggeredBy: • docker.socket
     Docs: https://docs.docker.com
    Main PID: 6354 (dockerd)
      Tasks: 7
     Memory: 32.8M
        CPU: 313ms
    CGroup: /system.slice/docker.service
            └─6354 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
```

Comprobamos que la imagen hello-world funciona correctamente:

docker run hello-world

```
root@debian12:/home/davidrl# docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
c1ec31eb5944: Pull complete
Digest: sha256:d211f485f2dd1dee407a80973c8f129f00d54604d2c90732e8e320e5038a0348
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.
```

1.2. Descarga imagen de ubuntu y accede a shell interactivo

Encuentra imagen disponible en docker hub de Ubuntu:

Usamos el siguiente comando para buscar una imagen de ubuntu en docker hub:

docker search ubuntu

```
root@debian12:/home/davidrl# docker search ubuntu
```

NAME	DESCRIPTION	STARS	OFFICIAL
ubuntu	Ubuntu is a Debian-based Linux operating sys...	17319	[OK]
ubuntu/squid	Squid is a caching proxy for the Web. Long-t...	99	
ubuntu/nginx	Nginx, a high-performance reverse proxy & we...	119	
ubuntu/cortex	Cortex provides storage for Prometheus. Long...	4	

Descárgala con comando pull.

Descargamos la imagen usando el comando:

docker pull ubuntu

```
root@debian12:/home/davidrl# docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
d1fbec07a2e5: Pull complete
Digest: sha256:ab64a8382e935382638764d8719362bb50ee418d944c1f3d26e0c99fae49a345
Status: Downloaded newer image for ubuntu:latest
docker.io/library/ubuntu:latest
root@debian12:/home/davidrl#
```

Comprueba las imágenes que tienes descargadas.

Comprobamos nuestras imágenes descargadas usando:

docker images

```
root@debian12:/home/davidrl# docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
ubuntu        latest    dc4c1391d370   3 days ago     78.1MB
hello-world    latest    d2c94e258dcb   17 months ago  13.3kB
root@debian12:/home/davidrl#
```

Accede a través de shell interactivo:

Arrancamos el contenedor y a través de los parámetros -it accedemos al Shell dentro del contenedor:

docker run -it Ubuntu

```
root@debian12:/home/davidrl# docker run -it ubuntu
root@f349ba8a8733:/# whoami
root
```

1.3. Configura entorno de nodejs

Instala nodejs y comprueba qué versión de nodejs ha quedado instalada.

Dentro de nuestro contenedor de Ubuntu, accedemos al Shell como hemos visto previamente y instalamos nodejs como lo haríamos en cualquier maquina:

Accedemos al Shell del contenedor Ubuntu y actualizamos repos:

```
root@debian12:/home/davidrl# docker run -it ubuntu
root@c2ac87e34388:/# apt-get update
Get:1 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:2 http://archive.ubuntu.com/ubuntu noble InRelease [256 kB]
Get:3 http://archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:4 http://archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:5 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Packages [372 kB]
Get:6 http://archive.ubuntu.com/ubuntu noble/main amd64 Packages [1808 kB]
```

Instalamos nodejs:

apt-get install nodejs

```
root@c2ac87e34388:/# apt-get install nodejs
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  ca-certificates libbrotli1 libcares2 libicu74 libnghttp2-14 libnode109 libuv1t64 node-acorn node-busboy node-
node-undici node-xtend nodejs-doc openssl
Suggested packages:
  npm
The following NEW packages will be installed:
  ca-certificates libbrotli1 libcares2 libicu74 libnghttp2-14 libnode109 libuv1t64 node-acorn node-busboy node-
node-undici node-xtend nodejs nodejs-doc openssl
0 upgraded, 15 newly installed, 0 to remove and 2 not upgraded.
Need to get 28.6 MB of archives.
After this operation, 111 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
```

Comprobamos la versión de nodejs instalada:

node -v

```
root@c2ac87e34388:/# node -v
v18.19.1
root@c2ac87e34388:/#
```

Salimos del contenedor:

exit

```
root@c2ac87e34388:/# exit
exit
root@debian12:/home/davidrl#
```

1.4 Guarda cambios en la imagen de ubuntu y haz push a tu repositorio de docker

Hasta ahora hemos instalado docker , hemos traído los contenedores de hello-world y el contenedor de Ubuntu, podemos listar todos los contenedores haciendo uso de:

docker ps

```
root@debian12:/home/davidrl# docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
```

Podemos observar que no sale nada en la lista, esto es debido a que docker ps solo muestra los contenedores que están activos, dado que nosotros hasta ahora solo hemos accedido al Shell de la maquina usando los parámetros `-it`, el contenedor solo se mantiene activo mientras el Shell esta abierto, por lo tanto cuando nosotros salimos de ese Shell, el contenedor detiene su ejecución.

Para ver todos los contenedores en nuestra maquina independientemente de su estado usaremos el parámetro `-a`

`docker ps -a`

```
root@debian12:/home/davidrl# docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
f349ba8a8733	ubuntu	"/bin/bash"	41 minutes ago	Exited (0) 40 minutes ago		flamboyant_mirzakhani
c3eb270fdec7	hello-world	"/hello"	2 hours ago	Exited (0) 2 hours ago		compassionate_cannon

Para simplemente para manipular los contenedores de docker existentes en nuestra maquina podemos usar los siguientes comandos:

Arrancar contenedor:

`docker start [nombreContenedor | idContenedor]`

Detener contenedor:

`docker stop [nombreContenedor | idContenedor]`

```
root@debian12:/home/davidrl# docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
f349ba8a8733	ubuntu	"/bin/bash"	41 minutes ago	Exited (0) 40 minutes ago		flamboyant_mirzakhani
c3eb270fdec7	hello-world	"/hello"	2 hours ago	Exited (0) 2 hours ago		compassionate_cannon

```
root@debian12:/home/davidrl# docker start flamboyant_mirzakhani
flamboyant_mirzakhani
```

Ahora si ejecutamos `docker ps` veremos solo el contenedor que esta activo:

```
root@debian12:/home/davidrl# docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
f349ba8a8733	ubuntu	"/bin/bash"	44 minutes ago	Up About a minute		flamboyant_mirzakhani

```
root@debian12:/home/davidrl#
```

Para borrar los contenedores de nuestra maquina usaremos el siguiente comando:

`docker rm [nombreContenedor | idContenedor]`

Esto ultimo nos lleva a una pregunta, al ser docker una herramienta tan flexible, que nos permite hacer "pull" de nuestros contenedores desde diferentes entornos, es mas que probable que nos interese poder hacer pull de nuestros propios contenedores en lugar de las imágenes del docker hub, en lugar de simplemente copiar los contenedores de una maquina a otra, los contenedores solo almacenan los cambios que se hacen de manera local asi que docker nos permite subir nuestros contenedores, para poder hacer uso de ellos desde donde queramos, para ello debemos registrarnos en [Docker Hub](https://hub.docker.com/) usaremos nuestro usuario en Docker hub para subir nuestros contenedores personalizados.

Para poder subir nuestros contenedores, primero debemos crear una imagen de nuestro contenedor, usaremos el siguiente comando:

docker commit -m "Descripcion de la imagen" -a "Nombre de autor" [nombreContenedor | idContenedor]/nombreImagen

```
root@debian12:/home/davidrl# docker commit -m "ubuntu con nodejs" -a "davidrl" f349ba8a8733 root/ubuntuodejs
sha256:da5f7963472921936d90f9c5f39a0074c89ad7f1c630bc6ce19bc4ca78d7df6d
root@debian12:/home/davidrl#
```

Ahora podemos listar las imágenes que hemos creado localmente:

docker images

```
root@debian12:/home/davidrl# docker images
REPOSITORY          TAG         IMAGE ID      CREATED        SIZE
root/ubuntuodejs    latest      da5f79634729  46 seconds ago  78.1MB
ubuntu              latest      dc4c1391d370  3 days ago    78.1MB
hello-world         latest      d2c94e258dcb  17 months ago  13.3kB
root@debian12:/home/davidrl#
```

Ahora debemos hacer login en DockerHub desde la maquina usando el siguiente comando:

docker login -u usuario

```
root@debian12:/home/davidrl# docker login -u davidrldq
Password:
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credential-stores

Login Succeeded
root@debian12:/home/davidrl#
```

Ahora podemos hacer push de nuestra imagen a nuestro repositorio:

*Hay que tener en cuenta que si nuestro nombre de usuario en docker es distinto del nombre de usuario que creo el contenedor antes de hacer el push debemos hacer tag de esa imagen a nuestro nombre de usuario de Docker Hub de la siguiente manera:

docker tag usuarioLocal/nombreContenedor usuarioDocker/nombreContenedor

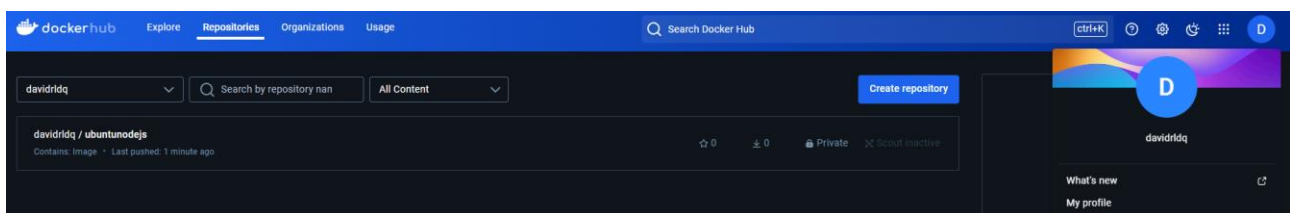
```
root@debian12:/home/davidrl# docker tag root/ubuntuodejs davidrldq/ubuntuodejs
root@debian12:/home/davidrl#
```

Ahora podemos subir nuestra imagen a nuestro repositorio en Docker Hub:

docker push nombreUsuario/nombreImagenDocker

```
root@debian12:/home/davidrl# docker push davidrldq/ubuntuodejs
Using default tag: latest
The push refers to repository [docker.io/davidrldq/ubuntuodejs]
8f3a58ad2658: Pushed
fa0f10cc481e: Mounted from library/ubuntu
latest: digest: sha256:c760256a863e497ad7f477c19164d475dc6b9263b3c374362169d560e3f40d17 size: 736
root@debian12:/home/davidrl#
```

Si vamos a la pagina de nuestro Docker Hub , en nuestros repositorios podremos encontrar la imagen que hemos subido:



Ahora podríamos desde otra maquina distinta, hacer login en nuestra cuenta de Docker Hub en la terminal y hacer pull de nuestra maquina para hacer uso de ella.

2. Docker compose

2.1 Instalación de Docker compose

FUENTE: <https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-compose-on-ubuntu-22-04>

Ahora vamos a instalar Docker compose, la diferencia entre Docker y Docker compose es que en Docker podemos crear y gestionar contenedores de manera independiente, realizando la configuración en cada máquina, mientras que Docker Compose nos permite gestionar y configurar una agrupación de contenedores simplificando la administración de los mismos de manera centralizada, por lo que no nos haría falta ir contenedor a contenedor realizando las configuraciones para nuestro entorno, si no que disponemos de un archivo de configuración para el entorno en el cual ya definimos dicha configuración.

Primero vamos a crear una carpeta en la raíz de nuestro /home donde iremos guardando todo lo relativo a docker compose:

```
mkdir -p ~/.docker/cli-plugins/
```

```
root@debian12:/home/davidrl# mkdir -p ~/.docker/cli-plugins/
```

Una vez creada, descargamos el paquete de docker compose dentro de dicha carpeta, antes de ejecutar este comando comprobaremos cual es la ultima versión de docker compose [aquí](#), en nuestro caso la versión mas reciente a dia de hoy 13/10/2024 es la versión v2.29.7, ejecutamos entonces el siguiente comando:

```
curl -SL https://github.com/docker/compose/releases/download/v2.29.7/docker-compose-linux-x86_64 -o ~/.docker/cli-plugins/docker-compose
```

```
root@debian12:/home/davidrl# curl -SL https://github.com/docker/compose/releases/download/v2.29.7/docker-compose-linux-x86_64 -o ~/.docker/cli-plugins/docker-compose
```

% Total	% Received	% Xferd	Average Speed	Time	Time	Time	Current
			Dload	Upload	Total	Spent	Left
0	0	0	0	0	0	0	0
100	60.8M	100	60.8M	0	0	39.1M	0
					0:00:01	0:00:01	76.8M

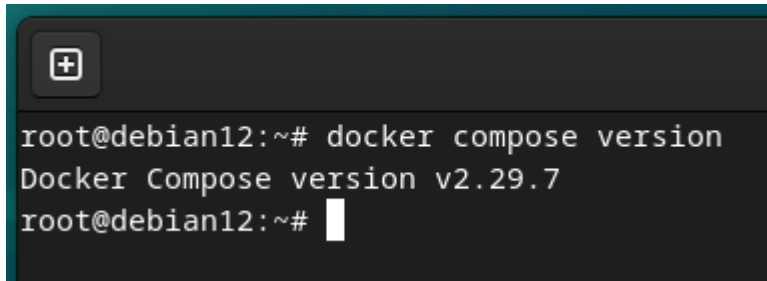
```
root@debian12:/home/davidrl#
```

Ahora damos permisos de ejecución a docker compose para poder usarlo:

```
chmod +x ~/.docker/cli-plugins/docker-compose
```

Y comprobamos nuestra versión de docker compose usando el siguiente comando:

```
docker compose version
```



```
root@debian12:~# docker compose version
Docker Compose version v2.29.7
root@debian12:~#
```

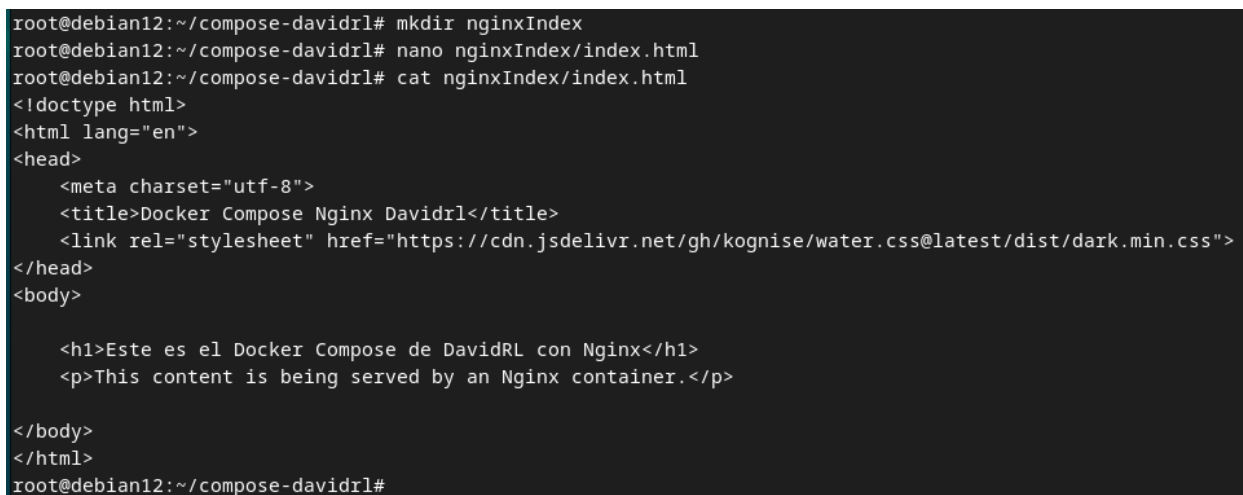
2.2 Despliega app de prueba sobre servidor web nginx con fichero docker-compose.yml

/* Sigue los pasos del tutorial, y muestra que la página web se carga correctamente en el puerto 8000 */

Ahora crearemos una nueva carpeta en la raíz de nuestro /home donde ira nuestro nuevo entorno de Docker compose para nginx:

```
mkdir ~/compose-davidrl
```

dentro de este directorio crearemos otro directorio llamado nginxIndex y dentro de este un archivo .html que servirá de index para nuestro nginx:



```
root@debian12:~/compose-davidrl# mkdir nginxIndex
root@debian12:~/compose-davidrl# nano nginxIndex/index.html
root@debian12:~/compose-davidrl# cat nginxIndex/index.html
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Docker Compose Nginx Davidrl</title>
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/gh/kognise/water.css@latest/dist/dark.min.css">
</head>
<body>

  <h1>Este es el Docker Compose de DavidRL con Nginx</h1>
  <p>This content is being served by an Nginx container.</p>

</body>
</html>
root@debian12:~/compose-davidrl#
```

Ahora crearemos el archivo de configuración de Docker compose, este archivo es un archivo .yaml donde podremos configurar todos los servicios que corren en nuestro entorno de docker compose:

```
root@debian12:~/compose-davidrl# nano docker-compose.yml
root@debian12:~/compose-davidrl# cat docker-compose.yml
services:
  web:
    image: nginx:alpine
    ports:
      - "8000:80"
    volumes:
      - ./nginxIndex:/usr/share/nginx/html
root@debian12:~/compose-davidrl#
```

Hay varias cosas a tener en cuenta en este archivo de configuración, la disposición de los elementos, por ejemplo:

Web: "Nombre del servicio"

- Image : la imagen que tomara de Docker Hub
- Ports: "puerto Maquina Real : Puerto Contenedor"
- Volumnes:"carpetaMaquinaReal : Redireccion en el Contenedor"

Ahora podemos levantar el entorno que hemos creado haciendo uso del siguiente comando:

docker compose up -d

```
root@debian12:~/compose-davidrl# docker compose up -d
[+] Running 9/9
 ✓ web Pulled
   ✓ 43c4264eed91 Pull complete
   ✓ d1171b13e412 Pull complete
   ✓ 596d53a7de88 Pull complete
   ✓ f99ac9ba1313 Pull complete
   ✓ fd072e74e282 Pull complete
   ✓ 379754eea6a7 Pull complete
   ✓ 45eb579d59b2 Pull complete
   ✓ 472934715761 Pull complete
[+] Running 2/2
 ✓ Network compose-davidrl_default Created
 ✓ Container compose-davidrl-web-1 Started
root@debian12:~/compose-davidrl#
```

Podemos observar que Docker compose hizo pull de los elementos que necesitaba y levanto los servicios, podemos ver los entornos de Docker Compose que se están ejecutando usando el comando:

docker compose ps

```
root@debian12:~/compose-davidrl# docker compose ps
NAME                IMAGE             COMMAND             SERVICE  CREATED   STATUS    PORTS
compose-davidrl-web-1  nginx:alpine     "/docker-entrypoint..."  web      2 minutes ago  Up 2 minutes  0.0.0.0:8000->80/tcp, [::]:8000->80/tcp
root@debian12:~/compose-davidrl#
```

Ahora comprobamos que la pagina esta levantado en el puerto indicado accediendo a localhost:8000



2.3. Pila LAMP con Docker compose

/*Configura pila LAMP y muestra correcto acceso conforme a alguno de los tutoriales:

FUENTES:

<https://github.com/sprintcube/docker-compose-lamp>

https://www.crashell.com/estudio/apache_php_mysql_y_phpmyadmin_con_docker_lamp

*/

Ahora vamos a levantar una pila LAMP con Docker compose, primero crearemos un nuevo directorio donde estará nuestro entorno:

```
root@debian12:~# mkdir compose-lamp-davidrl
```

Como me estoy basando en [esta guía](#) para hacer esta practica, tal y como me indica vamos a clonar el repositorio donde están todos los elementos necesarios en el directorio que hemos creado, usamos el siguiente comando:

git clone https://github.com/sprintcube/docker-compose-lamp.git

```
root@debian12:~/compose-lamp-davidrl# git clone https://github.com/sprintcube/docker-compose-lamp.git
Cloning into 'docker-compose-lamp'...
remote: Enumerating objects: 1025, done.
remote: Counting objects: 100% (309/309), done.
remote: Compressing objects: 100% (77/77), done.
remote: Total 1025 (delta 273), reused 234 (delta 231), pack-reused 716 (from 1)
Receiving objects: 100% (1025/1025), 258.18 KiB | 1.90 MiB/s, done.
Resolving deltas: 100% (521/521), done.
root@debian12:~/compose-lamp-davidrl#
```

Tal y como se nos indica en la guía , copiamos el archivo sample.env y lo modificamos acorde a nuestras necesidades:

```
GNU nano 7.2 sample.env
# Please Note:
# In PHP Versions <= 7.4 MySQL8 is not supported due to lacking pdo support

# To determine the name of your containers
COMPOSE_PROJECT_NAME=lamp

# Possible values: php54, php56, php71, php72, php73, php74, php8, php81, php82, php83
PHPVERSION=php83
DOCUMENT_ROOT=./www
APACHE_DOCUMENT_ROOT=/var/www/html
VHOSTS_DIR=./config/vhosts
APACHE_LOG_DIR=./logs/apache2
PHP_INI=./config/php/php.ini
SSL_DIR=./config/ssl

# PHPMysqlAdmin
UPLOAD_LIMIT=512M
MEMORY_LIMIT=512M
```


En mi caso, yo ya tenia de instalaciones previas una base de datos en mariaDB y un servidor apache, por lo que si dejo la configuración por defecto, el despliegue fallara, por lo tanto iré al archivo docker-compose.yml y realizare cambios en los puertos para que funcione todo correctamente:

Primero el puerto del servicio web, configuramos el puerto 8002:

```
services:
  webserver:
    build:
      context: ./bin/${PHPVERSION}
    container_name: "${COMPOSE_PROJECT_NAME}-${PHPVERSION}"
    restart: "always"
    ports:
      - 8002:80
      - "${HOST_MACHINE_SECURE_HOST_PORT}:443"
    links:
      - database
    volumes:
      - $(DOCUMENT_ROOT - /www) : /var/www/html:rw
```

El puerto de la base de datos, pondremos el 3316:

```
database:
  build:
    context: "./bin/${DATABASE}"
  container_name: "${COMPOSE_PROJECT_NAME}-${DATABASE}"
  restart: "always"
  ports:
    - 127.0.0.1:3316:3306
  volumes:
    - $(MYSQL_INITDB_DIR - /config/initdb) : /docker-entrypoint-initdb.d
```

Lo configuramos también en el phpmyadmin:

```
phpmyadmin:
  image: phpmyadmin
  container_name: "${COMPOSE_PROJECT_NAME}-phpmyadmin"
  links:
    - database
  environment:
    PMA_HOST: database
    PMA_PORT: 3316
    PMA_USER: root
    PMA_PASSWORD: $(MYSQL_ROOT_PASSWORD)
```

Una vez hechos los cambios levantamos el entorno usando el comando :

docker compose up -d

```
root@debian12:~/compose-lamp-davidrl/docker-compose-lamp# docker compose up -d
WARN[0000] /root/compose-lamp-davidrl/docker-compose-lamp/docker-compose.yml: the attribute `ve
e it to avoid potential confusion
[+] Running 5/5
 ✓ Network lamp_default      Created
 ✓ Container lamp-redis      Started
 ✓ Container lamp-mysql8     Started
 ✓ Container lamp-phpmyadmin Started
 ✓ Container lamp-php83      Started
root@debian12:~/compose-lamp-davidrl/docker-compose-lamp#
```

Accedemos a <http://localhost:8002> para comprobar que funciona:

