

UNIDAD TRABAJO 5:

VIRTUALIZACIÓN. CONTENEDORES. DOCKER.

DESPLIEGUE DE APLICACIONES CON DOCKER-COMPOSE

Direcciones del repositorio

<https://github.com/murcieta-gva/Tareas.git>

<https://github.com/murcieta-gva/Tarea-JFMF.git>

Índice:

1. Despliegue de aplicaciones con <i>docker-compose</i>	3
1.1. Usos de Docker en Microservicios	3
1.2. ¿Qué es <i>Docker-Compose</i> ?	3
2. Arquitectura de microservicios con <i>compose.yaml</i>	4
2.1. ¿Qué es un fichero con extensión YAML?	4
2.2. Estructura de un fichero <i>compose.yaml</i>	4
1. Networks.....	5
2. Services	6
3. Volumes	6
4. Configs.....	6
5. Secrets.....	7
3. Comandos básicos de <i>docker-compose</i>	7
4. Practica lo aprendido	10
4.1. Entorno de desarrollo Node.js	10
6. Preparación del entorno <i>testing</i> con Node.js mediante comandos docker	10
7. Despliegue de un entorno de desarrollo Node.js mediante Dockerfile	12
8. Despliegue de un entorno de desarrollo Node.js mediante <i>docker-compose</i>	14
9. Entorno de desarrollo Node.js:slim (reducido)	15
4.2. Kali Linux y Juice Shop.....	17
5. Ejercicio propuesto	19
6. Referencias.....	16
6.1. Vídeos	16
6.2. Otras referencias:.....	16

1. Despliegue de aplicaciones con *docker-compose*

1.1. Usos de Docker en Microservicios

En el contexto de los microservicios, Docker se utiliza principalmente para la construcción, el despliegue y la gestión de aplicaciones. Algunos de los usos más comunes son:

- **Despliegue de aplicaciones:** Docker se utiliza para empaquetar y desplegar diferentes microservicios en contenedores separados, lo que facilita el proceso de despliegue y minimiza el riesgo de conflictos y errores.
- **Integración continua y entrega continua:** Docker se utiliza ampliamente en procesos de integración continua y entrega continua. Las imágenes de contenedores se construyen y prueban automáticamente, lo que facilita la implementación rápida y segura de nuevas funcionalidades.
- **Pruebas de aplicaciones:** Docker se utiliza para crear diferentes entornos de prueba para las aplicaciones, lo que ayuda a los desarrolladores a identificar y corregir errores antes de desplegar las aplicaciones en producción.
- **Infraestructura como código:** Docker se utiliza para automatizar la creación y el despliegue de infraestructura en diferentes entornos, lo que permite a los desarrolladores gestionar la infraestructura como código. Esto facilita la gestión de diferentes entornos de desarrollo, pruebas y producción para la misma aplicación.
- **Despliegue de aplicaciones en la nube:** Docker se utiliza ampliamente en la nube para desplegar aplicaciones en diferentes entornos. Los contenedores de Docker son portátiles y se pueden desplegar en diferentes proveedores de la nube con facilidad.

1.2. ¿Qué es *Docker-Compose*?

Docker Compose es una herramienta que permite definir y administrar aplicaciones multicontenedor. Permite describir la configuración de una aplicación utilizando un archivo YAML y luego utilizar ese archivo para crear y administrar los contenedores de la aplicación de manera fácil y reproducible. Docker Compose surge porque muchas aplicaciones requieren de más de un microservicio. Pero claro, **la idea de Docker es que únicamente ejecute un único microservicio por contenedor** y no varios a la vez.

Por tanto, en esta situación podemos hacer dos cosas:

1. **Ejecutar dos microservicios en un mismo Docker.** Lo bueno en esta situación es que no necesitamos aprender nada para ello, simplemente con Docker podríamos lograrlo. Sin embargo, esto es muy delicado, ya que, si uno de los dos servicios fallase, fallarían todos los servicios. Además, sería poco escalable, ya que se compartirían todos los recursos y se tendría que escalar todo a la vez, lo cual no tiene mucho sentido. Lo lógico sería que si, por ejemplo, una API que tenemos en Docker recibe muchas peticiones, escalemos solo esa API, y no todo lo demás.
2. **Utilizar Docker Compose:** con Docker Compose puedes crear varios contenedores y definir cómo quieres que se relacionen y cómo quieres que gestionen los datos que generan. De esta forma, si uno de los contenedores fallase, podrías (depende cómo lo configures) permitir que los otros servicios sigan funcionando.

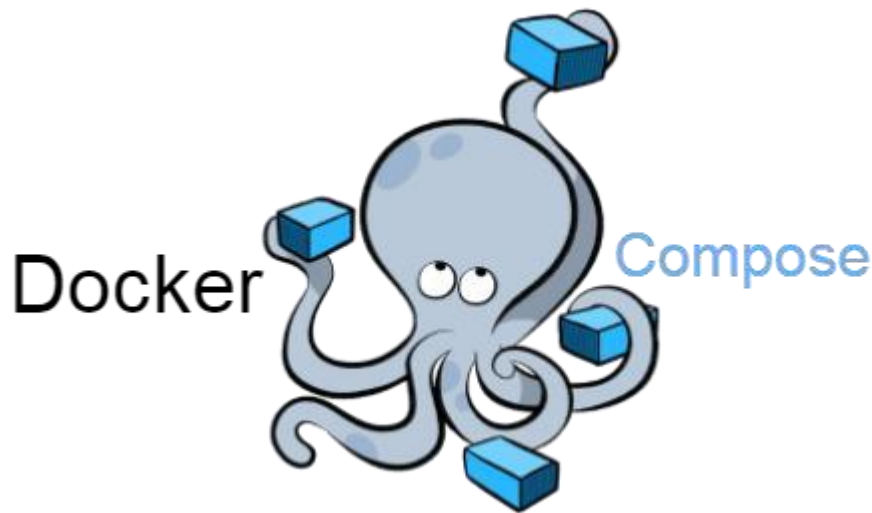


Figura 1. Logotipo docker-compose.

Con Docker Compose, se pueden definir los servicios, redes y volúmenes de una aplicación en un solo archivo, lo que facilita la configuración y el despliegue de la aplicación en diferentes entornos. Además, Docker Compose proporciona comandos para administrar los contenedores, como iniciar, detener y eliminar los servicios.

2. Arquitectura de microservicios con *compose.yaml*

2.1. ¿Qué es un fichero con extensión YAML?

YAML es un acrónimo que significa *Ain't Markup Language* (YAML no es un lenguaje de marcas). Se trata de un estándar de serialización de datos amigable para todos los lenguajes de programación. Más información en yaml.org.

Con **Compose** utilizaremos ficheros en formato **YAML**, que nos servirán para definir la configuración de la aplicación en cuestión. De esta manera podemos, con un solo comando, crear e iniciar los servicios configurados en estos ficheros.

2.2. Estructura de un fichero *compose.yaml*

Para crear un Docker Compose necesitamos crear un fichero *compose.yaml*. Este fichero es donde indicaremos qué servicios queremos que se ejecuten y de qué manera. Este fichero se estructura en los siguientes apartados:

- Versión (Opcional) • Servicios (Requerido).
- Redes.
- Volúmenes.
- Configs.
- Secrets.

1. Para entenderlo más fácilmente recurriremos a un fichero *compose.yaml* que despliega un servidor [Wordpress](#), junto con [MySQL](#) y [phpMyAdmin](#).



Figura 2.Contenedor Docker: MySQL, Wordpress y phpMyAdmin.

[Capturas de pantallas archivo compose.yml]

```

! compose.yml
1  networks:
2    ds-wordpress-6.1.1-net:
3      driver: bridge
4
5  services:
6    mysql:
7      image: mysql:5.7
8      container_name: ds-wordpress-6.1.1-mysql
9      tty: true
10     ports:
11       - "4208:3306"
12     volumes:
13       - "./var/libaclea/mysql:/var/lib/mysql"
14     environment:
15       MYSQL_ROOT_PASSWORD: 1234
16       MYSQL_DATABASE: wordpress
17       MYSQL_USER: miusuario
18       MYSQL_PASSWORD: mipassword
19     networks:
20       - ds-wordpress-6.1.1-net
21
22   server:
23     image: wordpress:latest
24     container_name: ds-wordpress-6.1.1
25     ports:
26       - "4282:80"
27     volumes:
28       - "./var/www/html:/var/www/html"
29     environment:
30       WORDPRESS_DB_USER: miusuario
31       WORDPRESS_DB_PASSWORD: mipassword
32       WORDPRESS_DB_NAME: wordpress
33       WORDPRESS_DB_HOST: ds-wordpress-6.1.1-mysql
34     depends_on:
35       - mysql
36     networks:
37       - ds-wordpress-6.1.1-net
38
39   phpmyadmin:
40     image: phpmyadmin/phpmyadmin
41     container_name: ds-phpmyadmin
42     ports:
43       - "4283:80"
44     environment:
45       PMA_HOST: ds-wordpress-6.1.1-mysql
46       MYSQL_ROOT_PASSWORD: 1234
47     depends_on:
48       - mysql
49     networks:
50       - ds-wordpress-6.1.1-net

```

].

1. Networks

Las redes es la capa que permite conectar a los servicios entre sí. Por ejemplo, en nuestro caso queremos que la aplicación pueda acceder a la base de datos.

El funcionamiento del **network** es el siguiente:

A cada servicio (**service**) indicamos el nombre del network al que pertenece, usando el parámetro **networks**. En la sección **networks** indicamos la configuración de cada uno de los networks.

- **driver**: especifica qué controlador se debe utilizar para esta red. **Compose** admite los siguientes controladores: **Docker Compose** admite varios tipos de controladores de red (**network drivers**) que se pueden utilizar para conectar los contenedores entre sí. A continuación, se describen los tipos de controladores de red admitidos por **Docker Compose**:
 - **bridge**: El controlador de red **bridge** es el predeterminado en Docker Compose. Crea una red interna en el host de Docker y asigna una interfaz de red virtual a cada contenedor conectado a esa red. Los contenedores en la misma red bridge pueden comunicarse entre sí utilizando sus nombres de host dentro de la red.
 - **host**: El controlador de red host utiliza la red del host de Docker en lugar de crear una red interna. Con este controlador, los contenedores comparten la misma interfaz de red que el host y pueden acceder directamente a los servicios que se ejecutan en el host sin necesidad de redireccionamiento de puertos.
 - **none**: El controlador de red **none** deshabilita completamente la red para un contenedor. Esto significa que el contenedor no tendrá acceso a ninguna red y estará completamente aislado.

2. Services

La definición de los servicios en **Docker Compose** es la única sección requerida y la más importante. En ella, definimos cada uno de los microservicios que vamos a ejecutar, con el nombre que queramos.

Además, para cada servicio se suelen definir los siguientes aspectos:

- **on-failure**: el contenedor se reinicia solo si hay un error.
- **unless-stopped**: el contenedor siempre se reinicia, hasta que el contenedor sea parado o eliminado.
- **volumes**: define rutas de *mount* o volúmenes que deben ser accesibles mediante otros servicios. En otras palabras, nos permite indicar qué carpetas queremos que sean copiadas de local a nuestro contenedor.
- **enviroment**: permite especificar las variables de entorno del contenedor.
- **depends_on**: permite definir dependencias de arranque y cierre entre diferentes servicios.

3. Volumes

Los volúmenes son formas de persistir la información, tal como hemos visto previamente en la sección de servicios. En este sentido, si queremos que varios contenedores accedan al mismo volumen, deberemos crear la sección `volumes`.

4. Configs

Al igual que la sección **volumes** permite la gestión y persistencia de datos, la sección **configs** sigue la misma idea, pero para la configuración de servicios. Supongamos, por ejemplo, que tienes un servidor Apache dentro de tu Docker Compose. Es probable que quieras cambiar la configuración de dicho servicio, pero claro, tener que hacer un **build** de la imagen cada vez que lo modificas no es algo óptimo.

En su lugar, a la hora de definir el servicio puedes indicar que se le debe aplicar una configuración y, en la sección **config** puedes explicar dicha configuración.

En este sentido, hay tres formas de definir la configuración:

- **file:** la configuración se crea a partir de un fichero en local.
- **external:** si se fija como **True** indica que la configuración ya se ha creado. Sirve para asegurarnos de que no se modifica algo que ya se ha configurado previamente.
- **name:** el nombre del **config** en Docker. Se puede fijar en caso de haber indicado `external: True`.

Ejemplo:

```
services:
  redis:
    image:
      redis:latest
    configs:
      -
        redis_conf configs:
          redis_conf:
            file: ./redis/redis.conf
```

5. Secrets

La sección **secrets** es una idea similar a la de **configs**, pero para permitir acceso a información sensible, tales como contraseñas o API Keys. Al igual que en el caso de **config**, se pueden definir **secrets** de varias formas:

- **file:** el **secret** es creado con el contenido de un fichero. **environment:** el **secret** se crea con el valor de una variable de entorno de tu sistema.
- **external:** si se fija como **True** indica que la configuración ya se ha creado. Sirve para asegurarnos de que no se modifica algo que ya se ha configurado previamente.
- **name:** el nombre del secret en Docker. Se puede fijar en caso de haber indicado `external: True`.

Ejemplo:

```
services:
  myapp:
    image:
      myapp:latest
    secrets:
      - api
    secrets:
      my_secret:
        file: ./my_secret.txt
```

3. Comandos básicos de *docker-compose*

Algunos comandos básicos de Docker Compose que te ayudarán a trabajar con tus archivos **docker-compose.yml** son:

2. **docker-compose up:** Este comando inicia todos los servicios definidos en el archivo **docker-compose.yml**. Si las imágenes de los servicios no están presentes, Docker Compose las construirá automáticamente. [\[Capturas de pantallas – A partir del fichero docker-compose.yml definido en los apartados anteriores vamos a lanzar los contenedores para trabajar con el servicio Wordpress\]](#)


```
MacBook-Pro-de-Jose:dockcom josefranciscumurciafuentes$ docker-compose up -d
[+] Running 1/3
  # phpmyadmin Pulling
  # server Pulling
  # mysql Error
  # mysql Error no matching manifest for linux/arm64/v8 in the manifest list entries: no match for platform in manifest: not found
Error response from daemon: no matching manifest for linux/arm64/v8 in the manifest list entries: no match for platform in manifest: not found
MacBook-Pro-de-Jose:dockcom josefranciscumurciafuentes$ docker-compose up -d
[+] Running 55/23
  # phpmyadmin Pulled
  # server Pulled
  # mysql Pulled
```

```
[+] Running 5/5
  # Network dockcom_ds-wordpress-6.1.1-net Created 0.0s
  # Container ds-wordpress-6.1.1-mysql Started 1.0s
  # Container ds-phpmyadmin Started 0.6s
  # Container ds-wordpress-6.1.1 Started 0.6s
  # phpmyadmin The requested image's platform (linux/amd64) does not match the detected host platform (linux/arm64/v8) and no specific platform was requested 0.0s
MacBook-Pro-de-Jose:dockcom josefranciscumurciafuentes$
```

El modificado el gestor de base de datos lo he cambiado por MariaDB debido a incompatibilidad de Mysql 5.7 con Procesadores Arm Apple Silicon].

```
docker-compose up -d
```

3. **docker-compose ps:** Este comando muestra el estado de los servicios definidos en el archivo **docker-compose.yml**. Proporciona información sobre los contenedores en ejecución, los puertos expuestos y el estado actual de cada servicio. [\[Capturas de pantallas](#)

```
[+] Running 5/5
  # Network dockcom_ds-wordpress-6.1.1-net Created 0.0s
  # Container ds-wordpress-6.1.1-mysql Started 1.0s
  # Container ds-phpmyadmin Started 0.6s
  # Container ds-wordpress-6.1.1 Started 0.6s
  # phpmyadmin The requested image's platform (linux/amd64) does not match the detected host platform (linux/arm64/v8) and no specific platform was requested 0.0s
MacBook-Pro-de-Jose:dockcom josefranciscumurciafuentes$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED     STATUS      PORTS                               NAMES
e1ea9a1601e7   wordpress:latest "docker-entrypoint.s..." 2 minutes ago Up 2 minutes 0.0.0.0:4282->80/tcp               ds-wordpress-6.1.1
595ebeb66d6f   phpmyadmin/phpmyadmin "/docker-entrypoint.s..." 2 minutes ago Up 2 minutes 0.0.0.0:4283->80/tcp               ds-phpmyadmin
5cb977398738   mariadb:10.5.8 "docker-entrypoint.s..." 2 minutes ago Up 2 minutes 0.0.0.0:4208->3306/tcp             ds-wordpress-6.1.1-mysql
MacBook-Pro-de-Jose:dockcom josefranciscumurciafuentes$
```

].

```
docker-compose ps
```

4. **docker-compose logs:** Este comando muestra los registros de los contenedores en ejecución. Puedes utilizar la opción **-f** para seguir los registros en tiempo real. [\[Capturas de pantallas](#)

```
josefranciscumurciafuentes@MacBook-Pro-de-Jose dockcom % docker-compose logs
ds-wordpress-6.1.1 | WordPress not found in /var/www/html - copying now...
ds-wordpress-6.1.1 | Complete! WordPress has been successfully copied to /var/www/html
ds-wordpress-6.1.1 | No 'wp-config.php' found in /var/www/html, but 'WORDPRESS_...' variables supplied; copying 'wp-config-docker.php' (WORDPRESS_DB_HOST
WORDPRESS_DB_NAME WORDPRESS_DB_PASSWORD WORDPRESS_DB_USER)
ds-wordpress-6.1.1 | AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 172.18.0.3. Set the 'ServerName' direc
tive globally to suppress this message
ds-wordpress-6.1.1 | AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 172.18.0.3. Set the 'ServerName' direc
tive globally to suppress this message
ds-wordpress-6.1.1 | [Sun Jan 26 16:46:21.775975 2025] [mpm_prefork:notice] [pid 1:tid 1] AH00163: Apache/2.4.62 (Debian) PHP/8.2.27 configured -- resumin
g normal operations
ds-wordpress-6.1.1 | [Sun Jan 26 16:46:21.776013 2025] [core:notice] [pid 1:tid 1] AH00094: Command line: 'apache2 -D FOREGROUND'
ds-phpmyadmin | AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 172.18.0.4. Set the 'ServerName' direc
tive globally to suppress this message
ds-phpmyadmin | AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 172.18.0.4. Set the 'ServerName' direc
tive globally to suppress this message
ds-wordpress-6.1.1-mysql | 2025-01-26 16:46:17+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 1:10.5.8+maria~focal started.
ds-wordpress-6.1.1-mysql | 2025-01-26 16:46:17+00:00 [Note] [Entrypoint]: Switching to dedicated user 'mysql'
ds-wordpress-6.1.1-mysql | 2025-01-26 16:46:17+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 1:10.5.8+maria~focal started.
ds-wordpress-6.1.1-mysql | 2025-01-26 16:46:17+00:00 [Note] [Entrypoint]: Initializing database files
ds-wordpress-6.1.1-mysql | PLEASE REMEMBER TO SET A PASSWORD FOR THE MariaDB root USER !
ds-wordpress-6.1.1-mysql | To do so, start the server, then issue the following commands:
ds-wordpress-6.1.1-mysql | 'usr/bin/mysqldadmin' -u root password 'new-password'
ds-wordpress-6.1.1-mysql | 'usr/bin/mysqldadmin' -u root -h password 'new-password'
ds-wordpress-6.1.1-mysql | Alternatively you can run:
ds-wordpress-6.1.1-mysql | 'usr/bin/mysql_secure_installation'
```



```

ds-wordpress-6.1.1-mysql | See the MariaDB Knowledgebase at https://mariadb.com/kb or the
ds-wordpress-6.1.1-mysql | MySQL manual for more instructions.
ds-wordpress-6.1.1-mysql |
ds-wordpress-6.1.1-mysql | Please report any problems at https://mariadb.org/jira
ds-wordpress-6.1.1-mysql |
ds-wordpress-6.1.1-mysql | The latest information about MariaDB is available at https://mariadb.org/.
ds-wordpress-6.1.1-mysql | You can find additional information about the MySQL part at:
ds-wordpress-6.1.1-mysql | https://dev.mysql.com
ds-wordpress-6.1.1-mysql | Consider joining MariaDB's strong and vibrant community:
ds-wordpress-6.1.1-mysql | https://mariadb.org/get-involved/
ds-phpmyadmin | [Sun Jan 26 16:46:18.285756 2025] [mpm_prefork:notice] [pid 1:tid 1] AH00163: Apache/2.4.62 (Debian) PHP/8.2.27 configured -- r
esuming normal operations
ds-phpmyadmin | [Sun Jan 26 16:46:18.286874 2025] [core:notice] [pid 1:tid 1] AH00094: Command line: 'apache2 -D FOREGROUND'
ds-wordpress-6.1.1-mysql | 2025-01-26 16:46:19+00:00 [Note] [Entrypoint]: Database files initialized
ds-wordpress-6.1.1-mysql | 2025-01-26 16:46:19+00:00 [Note] [Entrypoint]: Starting temporary server
ds-wordpress-6.1.1-mysql | 2025-01-26 16:46:19+00:00 [Note] [Entrypoint]: Waiting for server startup
ds-wordpress-6.1.1-mysql | 2025-01-26 16:46:19 0 [Note] mysqld (mysqld 10.5.8-MariaDB-1:10.5.8+maria-focal) starting as process 105 ...
ds-wordpress-6.1.1-mysql | 2025-01-26 16:46:19 0 [Warning] Setting lower_case_table_names=2 because file system for /var/lib/mysql/ is case insensitive
ds-wordpress-6.1.1-mysql | 2025-01-26 16:46:19 0 [Note] InnoDB: Using Linux native AIO
ds-wordpress-6.1.1-mysql | 2025-01-26 16:46:19 0 [Note] InnoDB: Uses event mutexes
ds-wordpress-6.1.1-mysql | 2025-01-26 16:46:19 0 [Note] InnoDB: Compressed tables use zlib 1.2.11
ds-wordpress-6.1.1-mysql | 2025-01-26 16:46:19 0 [Note] InnoDB: Number of pools: 1
ds-wordpress-6.1.1-mysql | 2025-01-26 16:46:19 0 [Note] InnoDB: Using ARMv8 crc32 + pmull instructions
ds-wordpress-6.1.1-mysql | 2025-01-26 16:46:19 0 [Note] mysqld: O_TMPFILE is not supported on /tmp (disabling future attempts)
ds-wordpress-6.1.1-mysql | 2025-01-26 16:46:19 0 [Note] InnoDB: Initializing buffer pool, total size = 134217728, chunk size = 134217728
ds-wordpress-6.1.1-mysql | 2025-01-26 16:46:19 0 [Note] InnoDB: Completed initialization of buffer pool
ds-wordpress-6.1.1-mysql | 2025-01-26 16:46:19 0 [Note] InnoDB: If the mysqld execution user is authorized, page cleaner thread priority can be changed. S
ee the man page of setpriority().
ds-wordpress-6.1.1-mysql | 2025-01-26 16:46:19 0 [Note] InnoDB: 128 rollback segments are active.
ds-wordpress-6.1.1-mysql | 2025-01-26 16:46:19 0 [Note] InnoDB: Creating shared tablespace for temporary tables
ds-wordpress-6.1.1-mysql | 2025-01-26 16:46:19 0 [Note] InnoDB: Setting file './ibtmp1' size to 12 MB. Physically writing the file full; Please wait ...
ds-wordpress-6.1.1-mysql | 2025-01-26 16:46:19 0 [Note] InnoDB: File './ibtmp1' size is now 12 MB.
ds-wordpress-6.1.1-mysql | 2025-01-26 16:46:19 0 [Note] InnoDB: 10.5.8 started; log sequence number 45118; transaction id 20
ds-wordpress-6.1.1-mysql | 2025-01-26 16:46:19 0 [Note] Plugin 'FEEDBACK' is disabled.
ds-wordpress-6.1.1-mysql | 2025-01-26 16:46:19 0 [Note] InnoDB: Loading buffer pool(s) from /var/lib/mysql/ib_buffer_pool
ds-wordpress-6.1.1-mysql | 2025-01-26 16:46:19 0 [Note] InnoDB: Buffer pool(s) load completed at 250126 16:46:19
ds-wordpress-6.1.1-mysql | 2025-01-26 16:46:19 0 [Warning] 'user' entry 'root@5cb977398738' ignored in --skip-name-resolve mode.
ds-wordpress-6.1.1-mysql | 2025-01-26 16:46:19 0 [Warning] 'proxies_priv' entry '@% root@5cb977398738' ignored in --skip-name-resolve mode.

ds-wordpress-6.1.1-mysql | 2025-01-26 16:46:22+00:00 [Note] [Entrypoint]: Stopping temporary server
ds-wordpress-6.1.1-mysql | 2025-01-26 16:46:22 0 [Note] mysqld (initiated by: root[root] @ localhost []) : Normal shutdown
ds-wordpress-6.1.1-mysql | 2025-01-26 16:46:22 0 [Note] Event Scheduler: Purging the queue. 0 events
ds-wordpress-6.1.1-mysql | 2025-01-26 16:46:22 0 [Note] InnoDB: FTS optimize thread exiting.
ds-wordpress-6.1.1-mysql | 2025-01-26 16:46:22 0 [Note] InnoDB: Starting shutdown...
ds-wordpress-6.1.1-mysql | 2025-01-26 16:46:22 0 [Note] InnoDB: Dumping buffer pool(s) to /var/lib/mysql/ib_buffer_pool
ds-wordpress-6.1.1-mysql | 2025-01-26 16:46:22 0 [Note] InnoDB: Buffer pool(s) dump completed at 250126 16:46:22
ds-wordpress-6.1.1-mysql | 2025-01-26 16:46:23 0 [Note] InnoDB: Shutdown completed; log sequence number 45130; transaction id 21
ds-wordpress-6.1.1-mysql | 2025-01-26 16:46:23 0 [Note] InnoDB: Removed temporary tablespace data file: "ibtmp1"
ds-wordpress-6.1.1-mysql | 2025-01-26 16:46:23 0 [Note] mysqld: Shutdown complete

ds-wordpress-6.1.1-mysql |
ds-wordpress-6.1.1-mysql | 2025-01-26 16:46:23+00:00 [Note] [Entrypoint]: Temporary server stopped
ds-wordpress-6.1.1-mysql |
ds-wordpress-6.1.1-mysql | 2025-01-26 16:46:23+00:00 [Note] [Entrypoint]: MySQL init process done. Ready for start up.
ds-wordpress-6.1.1-mysql |
ds-wordpress-6.1.1-mysql | 2025-01-26 16:46:23 0 [Note] mysqld (mysqld 10.5.8-MariaDB-1:10.5.8+maria-focal) starting as process 1 ...
ds-wordpress-6.1.1-mysql | 2025-01-26 16:46:23 0 [Warning] Setting lower_case_table_names=2 because file system for /var/lib/mysql/ is case insensitive
ds-wordpress-6.1.1-mysql | 2025-01-26 16:46:23 0 [Note] InnoDB: Using Linux native AIO
ds-wordpress-6.1.1-mysql | 2025-01-26 16:46:23 0 [Note] InnoDB: Uses event mutexes
ds-wordpress-6.1.1-mysql | 2025-01-26 16:46:23 0 [Note] InnoDB: Compressed tables use zlib 1.2.11
ds-wordpress-6.1.1-mysql | 2025-01-26 16:46:23 0 [Note] InnoDB: Number of pools: 1
ds-wordpress-6.1.1-mysql | 2025-01-26 16:46:23 0 [Note] InnoDB: Using ARMv8 crc32 + pmull instructions
ds-wordpress-6.1.1-mysql | 2025-01-26 16:46:23 0 [Note] mysqld: O_TMPFILE is not supported on /tmp (disabling future attempts)
ds-wordpress-6.1.1-mysql | 2025-01-26 16:46:23 0 [Note] InnoDB: Initializing buffer pool, total size = 134217728, chunk size = 134217728
ds-wordpress-6.1.1-mysql | 2025-01-26 16:46:23 0 [Note] InnoDB: Completed initialization of buffer pool
ds-wordpress-6.1.1-mysql | 2025-01-26 16:46:23 0 [Note] InnoDB: If the mysqld execution user is authorized, page cleaner thread priority can be changed. !
ee the man page of setpriority().
ds-wordpress-6.1.1-mysql | 2025-01-26 16:46:23 0 [Note] InnoDB: 128 rollback segments are active.
ds-wordpress-6.1.1-mysql | 2025-01-26 16:46:23 0 [Note] InnoDB: Creating shared tablespace for temporary tables
ds-wordpress-6.1.1-mysql | 2025-01-26 16:46:23 0 [Note] InnoDB: Setting file './ibtmp1' size to 12 MB. Physically writing the file full; Please wait ...
ds-wordpress-6.1.1-mysql | 2025-01-26 16:46:23 0 [Note] InnoDB: File './ibtmp1' size is now 12 MB.
ds-wordpress-6.1.1-mysql | 2025-01-26 16:46:23 0 [Note] InnoDB: 10.5.8 started; log sequence number 45130; transaction id 20
ds-wordpress-6.1.1-mysql | 2025-01-26 16:46:23 0 [Note] Plugin 'FEEDBACK' is disabled.
ds-wordpress-6.1.1-mysql | 2025-01-26 16:46:23 0 [Note] InnoDB: Loading buffer pool(s) from /var/lib/mysql/ib_buffer_pool
ds-wordpress-6.1.1-mysql | 2025-01-26 16:46:23 0 [Note] InnoDB: Buffer pool(s) load completed at 250126 16:46:23
ds-wordpress-6.1.1-mysql | 2025-01-26 16:46:23 0 [Note] Server socket created on IP: '::'.
ds-wordpress-6.1.1-mysql | 2025-01-26 16:46:23 0 [Warning] 'proxies_priv' entry '@% root@5cb977398738' ignored in --skip-name-resolve mode.
ds-wordpress-6.1.1-mysql | 2025-01-26 16:46:23 0 [Note] Reading of all Master_info entries succeeded
ds-wordpress-6.1.1-mysql | 2025-01-26 16:46:23 0 [Note] Added new Master_info '' to hash table
ds-wordpress-6.1.1-mysql | 2025-01-26 16:46:23 0 [Note] mysqld: ready for connections.
ds-wordpress-6.1.1-mysql | Version: '10.5.8-MariaDB-1:10.5.8+maria-focal' socket: '/run/mysqld/mysqld.sock' port: 3306 mariadb.org binary distribution
josefranciscomurciafuentes@MacBook-Pro-de-Jose dockcom %

```

].

docker-compose logs

5. **docker-compose down:** Este comando detiene y elimina todos los contenedores, redes y volúmenes creados por **docker-compose up**. Es útil para limpiar el entorno y asegurarse de que todos los recursos relacionados con el archivo **docker-compose.yml** se eliminen correctamente. [\[Capturas de pantallas\]](#)

```

josefranciscomurciafuentes@MacBook-Pro-de-Jose dockcom % docker-compose down
[+] Running 4/3
✔ Container ds-wordpress-6.1.1          Removed          1.3s
✔ Container ds-phpmyadmin              Removed          1.3s
✔ Container ds-wordpress-6.1.1-mysql    Removed          1.1s
✔ Network dockcom_ds-wordpress-6.1.1-net Removed          0.6s
josefranciscomurciafuentes@MacBook-Pro-de-Jose dockcom %

```

].

docker-compose down

4. Practica lo aprendido

4.1. Entorno de desarrollo Node.js

6. Preparación del entorno *testing* con Node.js mediante comandos docker

En primer lugar, aprenderemos como preparar un entorno de desarrollo Node.js sobre un sistema operativo Linux empleado comandos Docker. Los pasos que debemos seguir son:

6. Descarga de la imagen del entorno **Nodejs**. *[Capturas de pantallas para toda la sección:]*

```
josefranciscomurciafuentes@MacBook-Pro-de-Jose dockcom % docker pull node:latest
latest: Pulling from library/node
1d94403e0249: Download complete
94c5996c7a64: Download complete
7e80f679e1b1: Download complete
d22b85d68f8a: Download complete
936252136b92: Download complete
b1c7ed95aa40: Download complete
5e8de8f36c90: Download complete
e474a4a4cbbf: Download complete
Digest: sha256:3b73c4b366d490f76908dda253bb4516bbb3398948fd880d8682c5ef16427eca
Status: Downloaded newer image for node:latest
docker.io/library/node:latest
josefranciscomurciafuentes@MacBook-Pro-de-Jose dockcom %
```

].

```
docker pull node:latest
```

7. Comprobamos que la imagen del sistema operativo se ha descargado correctamente.

```
josefranciscomurciafuentes@MacBook-Pro-de-Jose dockcom % docker pull node:latest
latest: Pulling from library/node
1d94403e0249: Download complete
94c5996c7a64: Download complete
7e80f679e1b1: Download complete
d22b85d68f8a: Download complete
936252136b92: Download complete
b1c7ed95aa40: Download complete
5e8de8f36c90: Download complete
e474a4a4cbbf: Download complete
Digest: sha256:3b73c4b366d490f76908dda253bb4516bbb3398948fd880d8682c5ef16427eca
Status: Downloaded newer image for node:latest
docker.io/library/node:latest
josefranciscomurciafuentes@MacBook-Pro-de-Jose dockcom % docker images
REPOSITORY          TAG         IMAGE ID      CREATED       SIZE
phpmyadmin/phpmyadmin latest      95e01f723b5e  2 days ago   814MB
node                 latest      3b73c4b366d4  4 days ago   1.6GB
```

```
docker images
```

8. Ejecutamos un contenedor Docker a partir de la imagen de **Nodejs** descargada.

```
josefranciscomurciafuentes@MacBook-Pro-de-Jose dockcom % cd app
josefranciscomurciafuentes@MacBook-Pro-de-Jose app % docker run -it --entrypoint bash --name jfmf-my-javascript-app
-p 8080:8080 -v ${PWD}/my-javascript-app:/app -w /app node:latest

root@9652f53ad7c9:/app#
```

```
docker run -it --entrypoint bash --name jfmf-my-javascript-app -p 8080:8080 -v ${PWD}/my-javascript-app:/app
-w /app node:latest
```

9. Dentro del contenedor **/app#** ejecutamos **mocha**.

```
npm test.
```

10. En caso de que **mocha** no esté instalado lo instalamos previamente:

```
npm install mocha.
```

11. Lanzamos el servidor **Express** dentro del contenedor **Node**. En el contenedor **/app#** ejecutamos

[\[Capturas de pantallas para toda la sección:\]](#)

```
root@9652f53ad7c9:/app# npm init -y
Wrote to /app/package.json:
```

```
{
  "devDependencies": {
    "mocha": "^11.1.0",
    "request": "^2.88.2"
  },
  "name": "app",
  "version": "1.0.0",
  "main": "index.js",
  "dependencies": {
    "ajv": "^6.12.6",
    "ansi-colors": "^4.1.3",
    "ansi-regex": "^6.1.0",
    "ansi-styles": "^4.3.0",
    "anymatch": "^3.1.3",
    "argparse": "^2.0.1",
    "asn1": "^0.2.6",
    "assert-plus": "^1.0.0",
    "asynckit": "^0.4.0",
    "aws-sign2": "^0.7.0",
    "aws4": "^1.13.2",
    "balanced-match": "^1.0.2",
    "bcrypt-pbkdf": "^1.0.2",
    "binary-extensions": "^2.3.0",
    "brace-expansion": "^2.0.1",
    "braces": "^3.0.3",
    "browser-stdout": "^1.3.1",
    "camelcase": "^6.3.0",
    "caseless": "^0.12.0",
```



```

root@9652f53ad7c9:/app# node index.js &
[1] 261
root@9652f53ad7c9:/app# npm -v
11.0.0

[root@9652f53ad7c9:/app# npm run
Lifecycle scripts included in app@1.0.0:
  test
    mocha
root@9652f53ad7c9:/app#

```

```

josefranciscofuentes@MacBook-Pro-de-Jose ~ % docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS                    NAMES
9652f53ad7c9   node:latest    "bash"                  28 minutes ago Up 28 minutes  0.0.0.0:8080->8080/tcp   jfmf-my-javascript-app
josefranciscofuentes@MacBook-Pro-de-Jose ~ %

```

].

node src/index.js &, si todo ha ido bien debemos obtener la respuesta "Servidor escuchando en el puerto 8080"

7. Despliegue de un entorno de desarrollo Node.js mediante Dockerfile

12. En primer lugar, es necesario crear el fichero *Dockerfile*: .

[Capturas de pantallas de Dockerfile aplicación despliegue aplicación node:]

The screenshot shows the VS Code interface. On the left, the Explorer sidebar shows a project structure with folders 'Ejercicio' and 'node_modules', and a 'src' folder containing 'public', '.dockerignore', 'index.js', '.gitignore', 'compose-js-ap...', 'Dockerfile', 'package-lock.js...', 'package.json', and 'README.md'. The 'Dockerfile' file is selected. The main editor shows the content of the Dockerfile:

```

1 ARG VERSION=latest
2 FROM node:${VERSION:-latest}
3 RUN mkdir -p /usr/src/app
4 WORKDIR /usr/src/app
5 COPY package*.json ./
6 #Modificado el comando npm install
7 RUN npm install -g npm
8 COPY . .
9 EXPOSE 8080
10 CMD ["node", "src/index.js"]

```

].

Después de construida la imagen el siguiente paso es lanzar el contenedor asociado.

13. Podemos lanzar el servidor web y consultar la aplicación *Javascript* a través de un navegador web.

[Capturas de pantallas para toda la sección :

```
View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/8yqxa4w1zkebv7goziivbd24w
MacBook-Pro-de-Jose:~$ docker build -t mi-js-app-jfmf .
[+] Building 13.7s (12/12) FINISHED
=> [internal] load build definition from Dockerfile
=> [internal] load metadata for docker.io/library/node:latest
=> [internal] load .dockerignore
=> [internal] load context: 28
=> [1/6] FROM docker.io/library/node:latest@sha256:3b73c4b366d490f76908dda253bb4516bbb3398948fd880
=> [internal] load build context
=> [internal] load context: 214.89MB
=> [auth] library/node:pull token for registry-1.docker.io
=> CACHED [2/6] RUN mkdir -p /usr/src/app
=> CACHED [3/6] WORKDIR /usr/src/app
=> [4/6] COPY package*.json ./
=> [5/6] RUN npm install -g npm
=> [6/6] COPY . .
=> exporting to image
=> exporting layers
=> exporting manifest sha256:ed72d25720b0ac083f71577cef9e3215fd167b37c5fb288bbbc9935dfb0dc62a
=> exporting config sha256:6falc6537154e7bc4e80386a8edf17a234b240366849d62a8f9f8e2e9b1cc84a
=> exporting attestation manifest sha256:17739d9080fbb3dc987c8962573086d32213267e9228e0df19fd94
=> exporting manifest list sha256:5be06c06fe77a3820c62870661a894a0697f9c7cf632a4bad9fecce6bd5cb
=> naming to docker.io/library/mi-js-app-jfmf:latest
=> unpacking to docker.io/library/mi-js-app-jfmf:latest

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/y8ylqigdv27lw95oz715zlw5
MacBook-Pro-de-Jose:~$ docker run -p 8080:8080 mi-js-app-jfmf
Server in port 8080
```

].

```
docker run -it --name mi-js-app -p 8080:8080 mi-js-app
```

Paramos el contenedor que publica el servicio web mediante:

```
MacBook-Pro-de-Jose:~$ docker run -p 8080:8080 mi-js-app-jfmf
Server in port 8080
^C^Cdocker stop
^C
got 3 SIGTERM/SIGINTs, forcefully exiting
MacBook-Pro-de-Jose:~$ docker container stop mi-js-app-jfmf
mi-js-app-jfmf
MacBook-Pro-de-Jose:~$
```

```
docker container stop mi-js-app
```

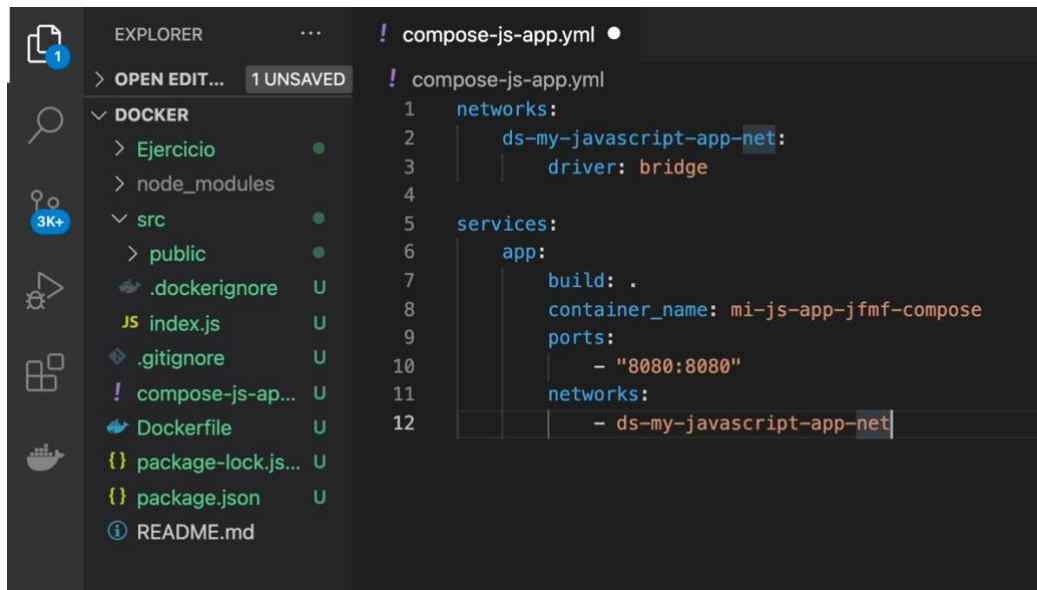
14. También es posible abrir una consola **bash** y ejecutar las pruebas dentro del contenedor.

```
MacBook-Pro-de-Jose:~$ docker run -it --entrypoint bash -p 8080:8080 mi-js-app-jfmf
root@3a9b725ce8b5:/usr/src/app# npm test
```

```
docker run -it --entrypoint bash -p 8080:8080 mi-js-app
npm test exit
```

8. Despliegue de un entorno de desarrollo Node.js mediante *dockercompose*

16. En primer lugar, es necesario crear el fichero **compose-js-app.yml**: . *[Capturas de pantallas para toda la sección y despliegue de la aplicación página web en Node:*



].

17. Una vez elaborado el fichero **compose-js-app.yml** lo lanzamos mediante la herramienta **docker-compose**:

[Capturas de pantallas para toda la sección:

```
MacBook-Pro-de-Jose:Docke josefranciscoturciafuentes$ docker-compose -f compose-js-app.yml up -d
[+] Running 1/1
✓ Container mi-js-app-jfmf-compose Started 0.4s
MacBook-Pro-de-Jose:Docke josefranciscoturciafuentes$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED      STATUS      PORTS                  NAMES
7c593f3c897f   docker-app "docker-entrypoint.s..." 48 seconds ago Up 47 seconds 0.0.0.0:8080->8080/tcp mi-js-app-jfmf-compos
```

```
docker-compose -f compose-js-app.yml up -d
```

18. Una vez compuesto el contenedor que contiene el **servidor Node** podemos consultar la aplicación Javascript a través de un navegador web. Para parar el servicio ejecutamos nuevamente **docker-compose**.

```
MacBook-Pro-de-Jose:Docke josefranciscoturciafuentes$ docker-compose -f compose-js-app.yml down
[+] Running 2/2
✓ Container mi-js-app-jfmf-compose Removed
✓ Network docker_ds-my-javascript-app-net Removed
MacBook-Pro-de-Jose:Docke josefranciscoturciafuentes$
```

```
docker-compose -f compose-js-app.yml down
```

9. Entorno de desarrollo Node.js:slim (reducido)

Cuando se lanzan contenedores en producción suele ser muy útil optimizar el tamaño de las imágenes y contenedores empleados con el objetivo que su tamaño sea lo más reducido posible.

19. Podemos observar cómo es posible esto estudiando el contenido del fichero *slim-jsapp.yml*.

[Capturas de pantallas:

```

1  networks:
2      ds-slim-javascript-app-net:
3          driver: bridge
4
5  services:
6      app:
7          image: mini-js-app
8          container_name: ds-slim-javascript-app
9          ports:
10             - "8080:8080"
11          networks:
12             - ds-slim-javascript-app-net

```

20. Debemos construir una imagen de tamaño reducido (extremadamente delgada), llamada *mini-js-app* empleando *Dockerfile* con el argumento *\$VERSION=slim*.

[Capturas de pantallas :

```

MacBook-Pro-de-Jose:~$ docker build --build-arg VERSION=slim -t docker-app .
[+] Building 14.8s (12/12) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 264B
=> [internal] load metadata for docker.io/library/node:slim
=> [auth] library/node:pull token for registry-1.docker.io
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [1/6] FROM docker.io/library/node:slim@sha256:f817b97de45c6e8046441c5ecef2f1c4fe45d31c3d2052fe82058ebe50fe7a94
=> => resolve docker.io/library/node:slim@sha256:f817b97de45c6e8046441c5ecef2f1c4fe45d31c3d2052fe82058ebe50fe7a94
=> => sha256:962260a6e0a6105f9483e64d69dc678d9e0bfc40395403157e206c4ab7688f8b 446B / 446B
=> => sha256:154a699a0e165763e5adb51681980f9fd83bdf817c522f4d6215afca7eb4a7 1.71MB / 1.71MB
=> => sha256:d88f6b68bae1deef63dfefcd580500bd9155b21f524c83d65c817136f42ad340 49.22MB / 49.22MB
=> => sha256:f9524bf8a101864d9f7c571c82e0a231cf10c637147370c2c86ac7e3c2e401d7 3.31kB / 3.31kB
=> => extracting sha256:f9524bf8a101864d9f7c571c82e0a231cf10c637147370c2c86ac7e3c2e401d7
=> => extracting sha256:d88f6b68bae1deef63dfefcd580500bd9155b21f524c83d65c817136f42ad340
=> => extracting sha256:54a699a0e165763e5adb51681980f9fd83bdf817c522f4d6215afca7eb4a7
=> => extracting sha256:962260a6e0a6105f9483e64d69dc678d9e0bfc40395403157e206c4ab7688f8b
=> [internal] load build context
=> => transferring context: 612.99kB
=> [2/6] RUN mkdir -p /usr/src/app
=> [3/6] WORKDIR /usr/src/app
=> [4/6] COPY package*.json ./
=> [5/6] RUN npm install --g npm
=> [6/6] COPY . .
=> => exporting to image
=> => exporting layers
=> => exporting manifest sha256:d31879aebdae39164e4cd44c6255a50b607a69d8e61a821147e71986780ce5e8
=> => exporting config sha256:c4c651dce168828efb6a119025f5e277d1933d97e20211899235b714d2c76a50
=> => exporting attestation manifest sha256:2eef37408fb081c2865ba4f2e5cb37c82fdcf8eaff7c11e1e84bb9060127f9d0
=> => exporting manifest list sha256:e76160feed8383ff31dd65848f5d869642ad930a324303d39c51c42d2658a027
=> => naming to docker.io/library/docker-app:latest
=> => unpacking to docker.io/library/docker-app:latest
View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/sazh7ti1pmdrgddov3w5ej
MacBook-Pro-de-Jose:~$

```

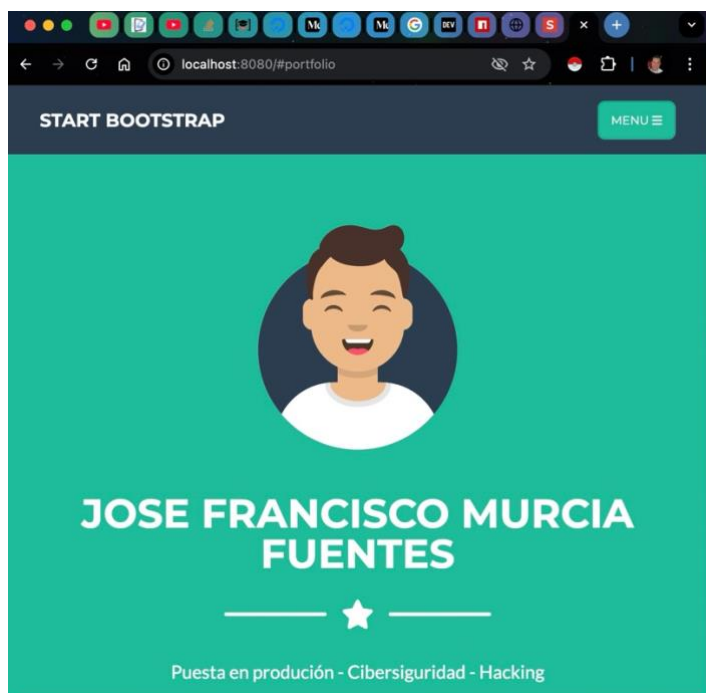
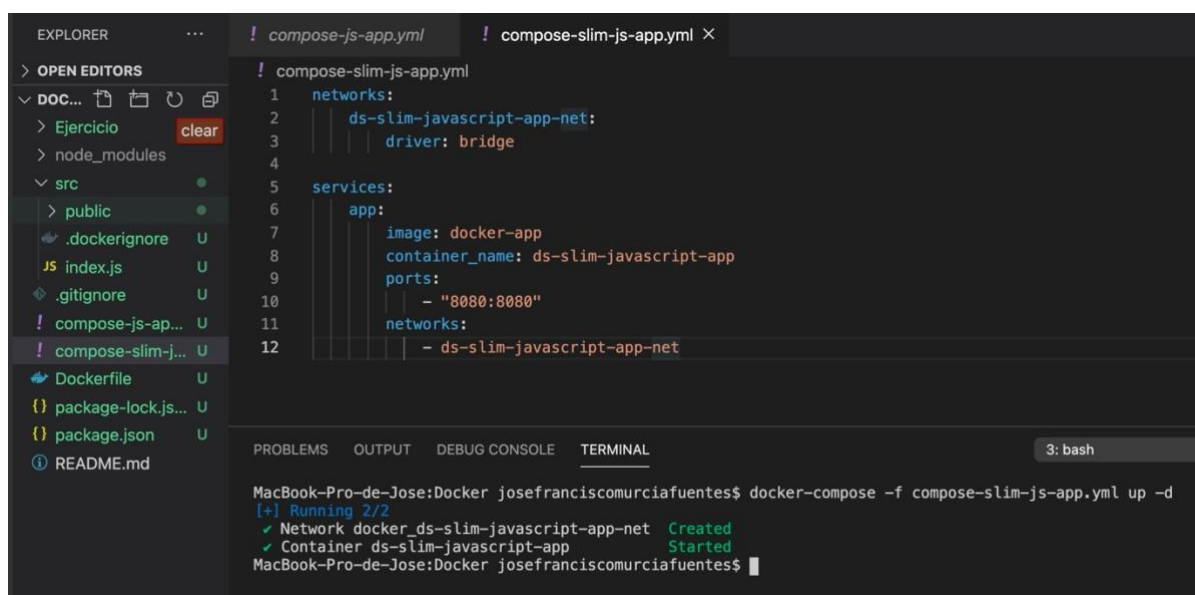

-Captura dode se muestran los tamaños de las imágenes:

```
MacBook-Pro-de-Jose:Docke josefranciscomurciafuentes$ docker-compose -f compose-js-app.yml down
[+] Running 2/2
✓ Container mi-js-app-jfmf-compose      Removed
✓ Network docker_ds-my-javascript-app-net Removed
MacBook-Pro-de-Jose:Docke josefranciscomurciafuentes$ docker images
REPOSITORY          TAG         IMAGE ID      CREATED        SIZE
docker-app           latest      4a6b76a731dc  31 minutes ago 1.93GB

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/sazhqn7ti1pmndrgddov3w5ej
MacBook-Pro-de-Jose:Docke josefranciscomurciafuentes$ docker images
REPOSITORY          TAG         IMAGE ID      CREATED        SIZE
docker-app           latest      e76160feed83  2 minutes ago  674MB
```

```
docker build --build-arg VERSION=slim -t mini-js-app .
```

21. Una vez construida la imagen mini-js-app, lanzaremos el contenedor:

[Capturas de pantallas

```
docker-compose -f slim-js-app.yml up -d.
```

22. Finalizada la sesión de trabajo paramos el contenedor.

[Capturas de pantallas:

```
MacBook-Pro-de-Jose:Docke josefranciscumurciafuentes$ docker-compose -f compose-slim-js-app.yml down
[+] Running 2/2
✓ Container ds-slim-javascript-app Removed
✓ Network docker_ds-slim-javascript-app-net Removed
```

].

```
docker-compose -f slim-js-app.yml down
```

4.2. Kali Linux y Juice Shop

23. Desplegaremos un laboratorio de pruebas con un contenedor KaliLinux y el sitio web para hacking Juice Shop.

[Capturas de pantallas despliegue contenedores:

```
MacBook-Pro-de-Jose:Compose josefranciscumurciafuentes$ docker-compose -f compose.yml up -d
[+] Running 2/3
✓ Network compose_red_peps Created 0.1s
✓ Container owasp-juiceshop-PePS Created 0.2s
# Container kali-PePS Creating 19.4s
Error response from daemon: failed to extract layer sha256:66bb15ef7b6d99bb3a3f7f75d11e7911cb3c6135769c7b7e23be681cd8abf6a: write /var/lib/desktop-containerd/daemon/io.containerd.snapshotter.v1.overlayfs/snapshots/849/fs/usr/share/ri/3.1.0/system/CGI/QueryExtension/multipart3f-1.ri: no space left on device: unknown
MacBook-Pro-de-Jose:Compose josefranciscumurciafuentes$ docker-compose -f compose.yml up -d
[+] Running 0/1
# Container compose-kali Creating 20.6s
Error response from daemon: failed to extract layer sha256:66bb15ef7b6d99bb3a3f7f75d11e7911cb3c6135769c7b7e23be681cd8abf6a: write /var/lib/desktop-containerd/daemon/io.containerd.snapshotter.v1.overlayfs/snapshots/852/fs/usr/share/ri/3.1.0/system/CGI/QueryExtension/read_from_cmdline-1.ri: no space left on device: unknown
MacBook-Pro-de-Jose:Compose josefranciscumurciafuentes$ docker-compose -f compose.yml up -d
[+] Running 0/1
# Container compose-kali Creating 29.2s
Error response from daemon: failed to extract layer sha256:10c4f08537c3ae8e76e8a2697b9a6d02e689986266db91a3e2e12ff4c37c7f52: write /var/lib/desktop-containerd/daemon/io.containerd.snapshotter.v1.overlayfs/snapshots/858/fs/usr/share/locale/hi/LC_MESSAGES/iso_639-3.mo: no space left on device: unknown
MacBook-Pro-de-Jose:Compose josefranciscumurciafuentes$ docker run -it --name kali-juice -p 3000:3000 compose-kali
(root@ 4b753efddd60)-[~]
```

```
Compose > ! compose.yml
1 networks:
2   red_peps:
3     driver: bridge
4
5 services:
6   juiceshop:
7     image: bkimminich/juice-shop
8     container_name: owasp-juiceshop-PePS
9     ports:
10      - 3000:3000
11     networks:
12      - red_peps
13 # Máquina kali Linux para Pentesting
14 kali:
15   build: .
16   container_name: compose-kali
17   depends_on:
18     - juiceshop
19   ports:
20     - 3389:3389
21   networks:
22     - red_peps
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: bash + ⓘ

```
MacBook-Pro-de-Jose:Compose josefranciscumurciafuentes$ docker-compose -f compose.yml up -d
[+] Running 2/2
✓ Container compose-kali Started 0.6s
✓ Container owasp-juiceshop-PePS Started 0.2s
MacBook-Pro-de-Jose:Compose josefranciscumurciafuentes$ docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED   STATUS    PORTS                               NAMES
3d8e3b0164b7   bkimminich/juice-shop               "/nodejs/bin/node /j-" 20 minutes ago   Up About a minute   0.0.0.0:3000->3000/tcp   owasp-juiceshop-PePS
MacBook-Pro-de-Jose:Compose josefranciscumurciafuentes$ docker images
REPOSITORY    TAG       IMAGE ID       CREATED      SIZE
compose-kali   latest   0e6c3273b3ba  39 minutes ago  7.26GB
docker-app     latest   5eacd339568b  20 hours ago  674MB
```

```

MacBook-Pro-de-Jose:Compose josefranciscomurciafuentes$ docker run -it --name kali-juice-2 -p 3389:3389 compose-kali
(root@5a96384cf425)~# arp -a
bash: arp: command not found

(root@5a96384cf425)~# service xrdp start
Starting Remote Desktop Protocol server: xrdp-sesman xrdpxrdp-sesman[26]: [INFO ] starting xrdp-sesman with pid 26
.

(root@5a96384cf425)~# xrdp-sesman[26]: [INFO ] Socket 11: AF_INET6 connection received from ::1 port 37814

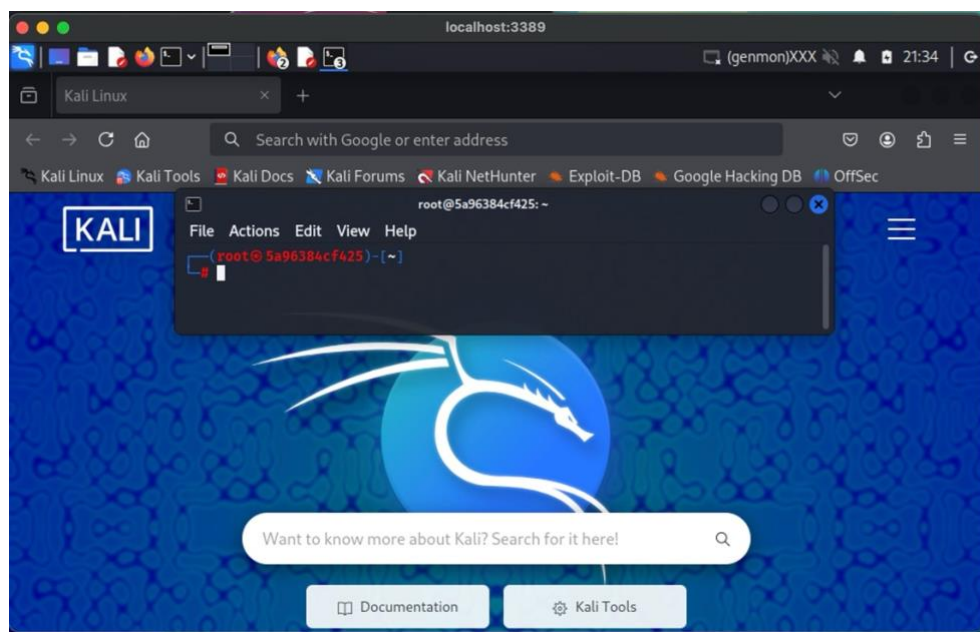
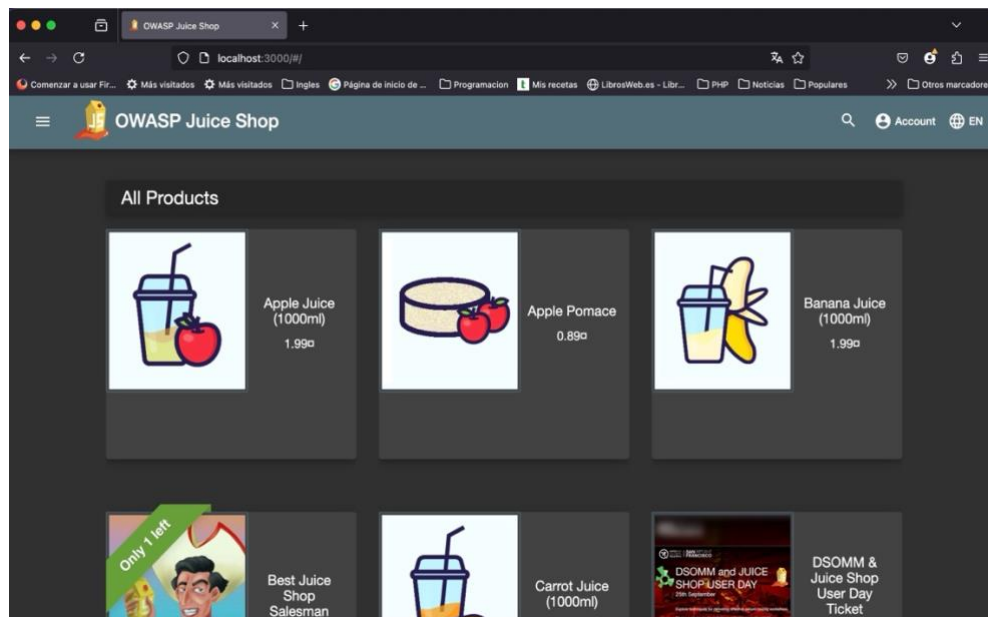
xrdp-sesman[26]: pam_unix(xrdp-sesman:auth): authentication failure; logname= uid=0 euid=0 tty=xrdp-sesman ruser= rhost= user=root
xrdp-sesman[26]: [ERROR] pam_authenticate failed: Authentication failure

xrdp-sesman[26]: [INFO ] AUTHFAIL: user=root ip=::ffff:172.17.0.1 time=1738008440

xrdp-sesman[26]: [ERROR] sesman_data_in: scp_process_msg failed

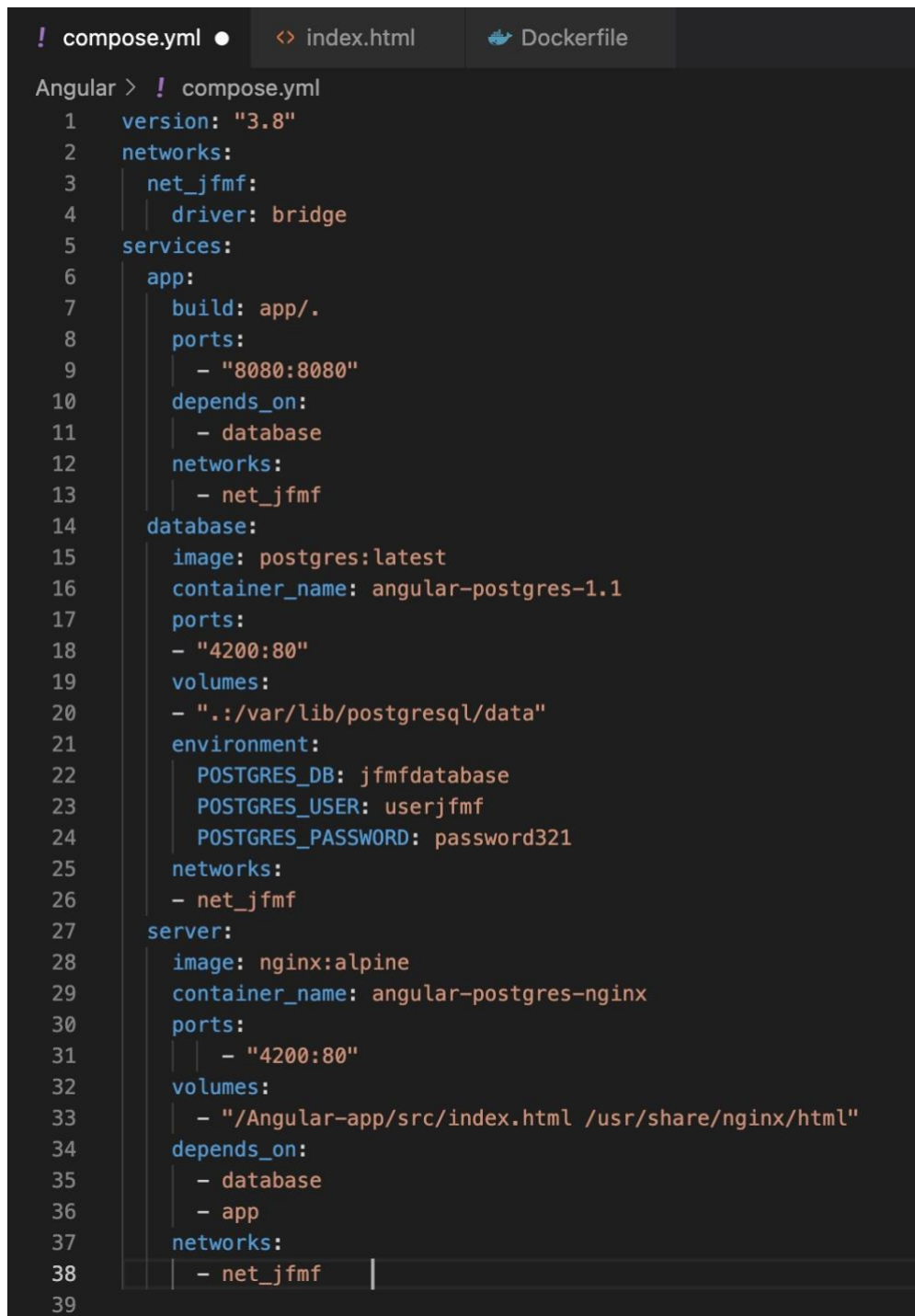
```

-Captura del sitio web Hacking Juice Shop:



5. Ejercicio propuesto

24. Aprende a configurar varios servicios con **Docker Compose** mediante ejercicios prácticos y ejemplos detallados. Prepara un archivo **docker-compose.yml** que defina los tres siguientes servicios: [\[Capturas de pantallas :\]](#)



```

! compose.yml ●  <> index.html  Dockerfile
Angular > ! compose.yml
1  version: "3.8"
2  networks:
3    net_jfmf:
4      driver: bridge
5  services:
6    app:
7      build: app/.
8      ports:
9        - "8080:8080"
10     depends_on:
11       - database
12     networks:
13       - net_jfmf
14   database:
15     image: postgres:latest
16     container_name: angular-postgres-1.1
17     ports:
18       - "4200:80"
19     volumes:
20       - ".: /var/lib/postgresql/data"
21     environment:
22       POSTGRES_DB: jfmfdatabase
23       POSTGRES_USER: userjfmf
24       POSTGRES_PASSWORD: password321
25     networks:
26       - net_jfmf
27   server:
28     image: nginx:alpine
29     container_name: angular-postgres-nginx
30     ports:
31       - "4200:80"
32     volumes:
33       - "/Angular-app/src/index.html /usr/share/nginx/html"
34     depends_on:
35       - database
36       - app
37     networks:
38       - net_jfmf
39

```

].

□ Frontend (Angular):

- Utiliza la imagen de Docker **nginx:alpine** para servir los archivos estáticos del proyecto Angular.

- Debe estar disponible en el puerto **4200** en el anfitrión (80 en el contenedor).
- Se asume que el proyecto ya está construido y los archivos estáticos se encuentran en una carpeta: **./dist/**, y deben montarse en el contenedor en **/usr/share/nginx/html**.
- Debe poder acceder al **backend** (depende del backend).

□ Backend (Spring Boot):

- Asume que dentro de una carpeta **./backend** existe un **Dockerfile** llamado **myBackend** que define una imagen **spring_app:latest**, con todo preparado para ser utilizada.
- Debe exponer el puerto **8080** (en anfitrión y contenedor).
- Debe poder acceder a la base de datos (depende de la base de datos).

□ Base de Datos (PostgreSQL):

- Utiliza la imagen oficial de PostgreSQL **postgres:latest**.
- Configura variables de entorno para el usuario (**POSTGRES_USER: userINICIALES**), la contraseña (**POSTGRES_PASSWORD: password321**), y el nombre de la base de datos (**POSTGRES_DB: inicialesdatabase**).
- Debe disponer de un volumen **postgres_data**, que se monta en **/var/lib/postgresql/data**. Este volumen debe ser definido dentro del **dockercompose**.

□ Escribe el archivo **docker-compose.yml** para la **versión 3.8**. Además, configure una red personalizada: **net_INICIALES**.

25. Inicia todos los servicios definidos en el archivo **docker-compose.yml**.

[Capturas de pantalla:

```

✓ Container angular-postgres-nginx Started 0.5s
MacBook-Pro-de-Jose:Angular josefranciscojurciafuentes$ docker-compose -f compose.yml up -d --build
WARN[0000] /Users/josefranciscojurciafuentes/Angular/compose.yml: the attribute 'version' is obsolete, it will be ignored, please remove it to avoid potential c
onfusion
[+] Building 1.4s (13/13) FINISHED docker:desktop-linux
=> [app internal] load build definition from Dockerfile 0.0s
=> => transferring dockerfile: 280B 0.0s
=> [app internal] load metadata for docker.io/library/eclipse-temurin:17-jdk-focal 1.2s
=> [app auth] library/eclipse-temurin:pull token for registry-1.docker.io 0.0s
=> [app internal] load .dockerignore 0.0s
=> => transferring context: 2B 0.0s
=> [app 1/6] FROM docker.io/library/eclipse-temurin:17-jdk-focal@sha256:dd9872e92e60db3ae2692b05b2c6bfe99d3f339ec0a36aba904ad70cedc2ea2c 0.0s
=> => resolve docker.io/library/eclipse-temurin:17-jdk-focal@sha256:dd9872e92e60db3ae2692b05b2c6bfe99d3f339ec0a36aba904ad70cedc2ea2c 0.0s
=> [app internal] load build context 0.0s
=> => transferring context: 4.69kB 0.0s
=> CACHED [app 2/6] WORKDIR /app 0.0s
=> CACHED [app 3/6] COPY .mvn/ .mvn 0.0s
=> CACHED [app 4/6] COPY mvnw pom.xml ./ 0.0s
=> CACHED [app 5/6] RUN ./mvnw dependency:go-offline 0.0s
=> CACHED [app 6/6] COPY src ./src 0.0s
=> [app] exporting to image 0.0s
=> => exporting layers 0.0s
=> => exporting manifest sha256:e09e60a94132bebcdbf0d9929eb44a22be9e2a3fea345fa6d9d23bcbbbf28b 0.0s
=> => exporting config sha256:11a3ea3d04210cc13aa376ef00de3eeae3399a5447c69f7a70baf1e9c9fd58a4 0.0s
=> => exporting attestation manifest sha256:321d09454cbeb8d4d152bec38a73d4d672988929f546eaba32360a72aee10a18 0.0s
=> => exporting manifest list sha256:0bbcf588abd19130e78c07433f0708cd4eba5330111a3c5b547281f85255d899 0.0s
=> => naming to docker.io/library/angular-app:latest 0.0s
=> => unpacking to docker.io/library/angular-app:latest 0.0s
=> [app] resolving provenance for metadata file 0.0s
[+] Running 3/3
✓ Container angular-postgres-1.1 Started 0.5s
✓ Container angular-postgres-nginx Started 1.0s
✓ Container angular-app-1 Started 0.6s
MacBook-Pro-de-Jose:Angular josefranciscojurciafuentes$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS                NAMES
5b2fe2164433   nginx:alpine   "/docker-entrypoint. ..." 28 minutes ago Up 12 seconds 0.0.0.0:4200->80/tcp   angular-postgres-nginx

```

].

26. Muestra el estado de los servicios definidos en el archivo **docker-compose.yml**.

[Capturas de pantallas:

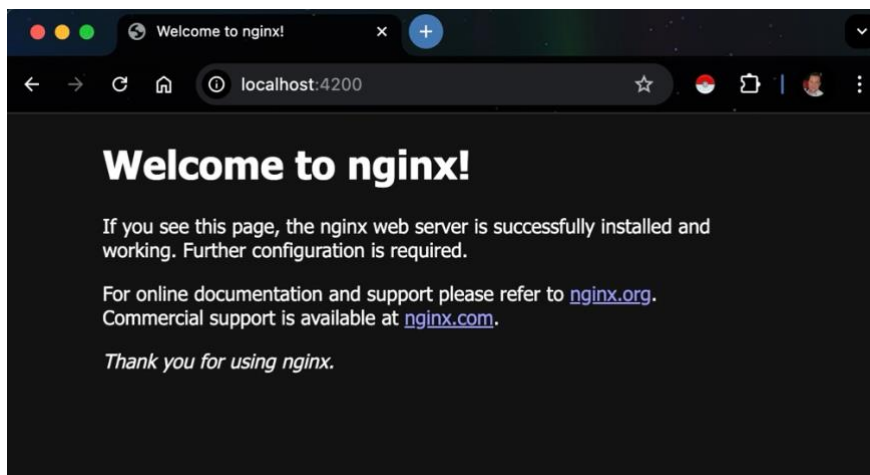
CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
5b2fe2164433	nginx:alpine	"/docker-entrypoint..."	About an hour ago	Up 31 minutes	0.0.0.0:4200->80/tcp	angular-postgres-nginx

].

27. Muestra los registros de los contenedores en ejecución. [Capturas de pantallas:

```
MacBook-Pro-de-Jose:Angular josefranciscumurciafuentes$ docker-compose logs
WARN[0000] /Users/josefranciscumurciafuentes/Angular/compose.yml: the attribute 'version' is obsolete, it will be ignored, please remove it to avoid potential confusion
app-1 [INFO] Scanning for projects...
app-1 [INFO]
app-1 [INFO] ----- com.jfmf:demo -----
app-1 [INFO] Building demo 0.0.1-SNAPSHOT
app-1 [INFO] from pom.xml
app-1 [INFO] [ jar ]
app-1 [INFO]
app-1 [INFO] >>> spring-boot:3.4.2:run (default-cli) > test-compile @ demo >>>
app-1 Downloading from central: https://repo.maven.apache.org/maven2/net/bytebuddy/byte-buddy/1.15.11/byte-buddy-1.15.11.jar (17 kB at 43 kB/s)
app-1 Downloading from central: https://repo.maven.apache.org/maven2/net/bytebuddy/byte-buddy-parent/1.15.11/byte-buddy-parent-1.15.11.jar (63 kB at 991 kB/s)
app-1 Downloading from central: https://repo.maven.apache.org/maven2/net/bytebuddy/byte-buddy-agent/1.15.11/byte-buddy-agent-1.15.11.jar (12 kB at 344 kB/s)
app-1 Downloading from central: https://repo.maven.apache.org/maven2/net/bytebuddy/byte-buddy/1.15.11/byte-buddy-1.15.11.jar (17 kB at 43 kB/s)
angular-postgres-1.1 | The files belonging to this database system will be owned by user "postgres".
angular-postgres-1.1 | This user must also own the server process.
angular-postgres-1.1 |
angular-postgres-1.1 | The database cluster will be initialized with locale "en_US.utf8".
angular-postgres-1.1 | The default database encoding has accordingly been set to "UTF8".
angular-postgres-1.1 | The default text search configuration will be set to "english".
angular-postgres-1.1 | /docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
angular-postgres-1.1 | /docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
angular-postgres-1.1 | /docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
angular-postgres-1.1 | 10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
angular-postgres-1.1 | 10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
angular-postgres-1.1 | /docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
angular-postgres-1.1 | /docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
angular-postgres-1.1 | /docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
angular-postgres-1.1 | /docker-entrypoint.sh: Configuration complete; ready for start up
angular-postgres-1.1 |
angular-postgres-1.1 | 2025/01/30 23:13:13 [notice] 1#1: using the "epoll" event method
angular-postgres-1.1 | Data page checksums are disabled.
angular-postgres-1.1 |
angular-postgres-1.1 | initdb: error: directory "/var/lib/postgresql/data" exists but is not empty
angular-postgres-1.1 | initdb: hint: If you want to create a new database system, either remove or empty the directory "/var/lib/postgresql/data" or run init
angular-postgres-1.1 | db with an argument other than "/var/lib/postgresql/data".
angular-postgres-1.1 | The files belonging to this database system will be owned by user "postgres".
angular-postgres-1.1 | This user must also own the server process.
angular-postgres-1.1 |
angular-postgres-1.1 | The database cluster will be initialized with locale "en_US.utf8".
angular-postgres-1.1 | The default database encoding has accordingly been set to "UTF8".
angular-postgres-1.1 | 2025/01/30 23:13:13 [notice] 1#1: nginx/1.27.3
angular-postgres-1.1 | https://repo.maven.apache.org/maven2/net/bytebuddy/byte-buddy/1.15.11/byte-buddy-1.15.11.jar (8.5 MB at 35 MB/s)
app-1 Downloading from central: https://repo.maven.apache.org/maven2/net/bytebuddy/byte-buddy-agent/1.15.11/byte-buddy-agent-1.15.11.jar (365 kB at 9.1 MB/s)
angular-postgres-1.1 | 2025/01/30 23:13:13 [notice] 1#1: built by gcc 13.2.1 20240309 (Alpine 13.2.1_git20240309)
app-1 [INFO]
angular-postgres-1.1 | The default text search configuration will be set to "english".
angular-postgres-1.1 |
angular-postgres-1.1 | The default text search configuration will be set to "english".
angular-postgres-1.1 | 2025/01/30 23:13:13 [notice] 1#1: OS: Linux 6.10.14-linuxkit
angular-postgres-1.1 | 2025/01/30 23:13:13 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
angular-postgres-1.1 | 2025/01/30 23:13:13 [notice] 1#1: start worker processes
angular-postgres-1.1 | 2025/01/30 23:13:13 [notice] 1#1: start worker process 30
angular-postgres-1.1 | 2025/01/30 23:13:13 [notice] 1#1: start worker process 31
angular-postgres-1.1 | 2025/01/30 23:13:13 [notice] 1#1: start worker process 32
angular-postgres-1.1 | 2025/01/30 23:13:13 [notice] 1#1: start worker process 33
angular-postgres-1.1 | 2025/01/30 23:13:13 [notice] 1#1: start worker process 34
angular-postgres-1.1 | 2025/01/30 23:13:13 [notice] 1#1: start worker process 35
angular-postgres-1.1 | 2025/01/30 23:13:13 [notice] 1#1: start worker process 36
angular-postgres-1.1 | 2025/01/30 23:13:13 [notice] 1#1: start worker process 37
app-1 [INFO] --- resources:3.3.1:resources (default-resources) @ demo ---
app-1 [INFO] Copying 1 resource from src/main/resources to target/classes
app-1 [INFO] Copying 0 resource from src/main/resources to target/classes
angular-postgres-1.1 |
angular-postgres-1.1 | Data page checksums are disabled.
angular-postgres-1.1 | 172.21.0.1 -- [30/Jan/2025:23:13:45 +0000] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/53
7.36 (KHTML, like Gecko) Chrome/132.0.0.0 Safari/537.36" "-"
angular-postgres-1.1 | initdb: error: directory "/var/lib/postgresql/data" exists but is not empty
angular-postgres-1.1 | 172.21.0.1 -- [30/Jan/2025:23:13:47 +0000] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/53
7.36 (KHTML, like Gecko) Chrome/132.0.0.0 Safari/537.36" "-"
angular-postgres-1.1 | initdb: hint: If you want to create a new database system, either remove or empty the directory "/var/lib/postgresql/data" or run init
db with an argument other than "/var/lib/postgresql/data".
angular-postgres-1.1 | The files belonging to this database system will be owned by user "postgres".
angular-postgres-1.1 | This user must also own the server process.
angular-postgres-1.1 |
angular-postgres-1.1 | The database cluster will be initialized with locale "en_US.utf8".
angular-postgres-1.1 | The default database encoding has accordingly been set to "UTF8".
angular-postgres-1.1 | 172.21.0.1 -- [30/Jan/2025:23:13:53 +0000] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/53
7.36 (KHTML, like Gecko) Chrome/132.0.0.0 Safari/537.36" "-"
angular-postgres-1.1 | The default text search configuration will be set to "english".
angular-postgres-1.1 | 172.21.0.1 -- [30/Jan/2025:23:13:54 +0000] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/53
7.36 (KHTML, like Gecko) Chrome/132.0.0.0 Safari/537.36" "-"
angular-postgres-1.1 | 172.21.0.1 -- [30/Jan/2025:23:13:54 +0000] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/53
7.36 (KHTML, like Gecko) Chrome/132.0.0.0 Safari/537.36" "-"
angular-postgres-1.1 | 2025/01/30 23:14:10 [notice] 1#1: signal 3 (SIGQUIT) received, shutting down
angular-postgres-1.1 | 2025/01/30 23:14:10 [notice] 32#32: gracefully shutting down
angular-postgres-1.1 | 2025/01/30 23:14:10 [notice] 30#30: gracefully shutting down
angular-postgres-1.1 | 2025/01/30 23:14:10 [notice] 33#33: gracefully shutting down
angular-postgres-1.1 | 2025/01/30 23:14:10 [notice] 31#31: gracefully shutting down
app-1 [INFO] compiler:3.13.0:compile (default-compile) @ demo ---
app-1 [INFO] Recompiling the module because of changed source code.
angular-postgres-1.1 | 2025/01/30 23:14:10 [notice] 35#35: gracefully shutting down
angular-postgres-1.1 | 2025/01/30 23:14:10 [notice] 37#37: gracefully shutting down
angular-postgres-1.1 | 2025/01/30 23:14:10 [notice] 36#36: gracefully shutting down
angular-postgres-1.1 | 2025/01/30 23:14:10 [notice] 32#32: exiting
angular-postgres-1.1 | 2025/01/30 23:14:10 [notice] 31#31: exiting
angular-postgres-1.1 | 2025/01/30 23:14:10 [notice] 35#35: exiting
angular-postgres-1.1 | 2025/01/30 23:14:10 [notice] 37#37: exiting
```

].

28. Prueba y ejecuta el entorno desplegado. [\[Capturas de pantalla\]](#)[\].](#)29. Detén y elimina todos los contenedores, redes y volúmenes creados por dockercompose up. [\[Capturas de pantallas :\]](#)

```
MacBook-Pro-de-Jose:Angular josefranciscumurciafuentes$ docker-compose stop
WARN[0000] /Users/josefranciscumurciafuentes/Angular/compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion
[+] Stopping 3/1
✓ Container angular-postgres-nginx Stopped 0.2s
✓ Container angular-app-1 Stopped 0.0s
✓ Container angular-postgres-1.1 Stopped 0.0s
MacBook-Pro-de-Jose:Angular josefranciscumurciafuentes$

MacBook-Pro-de-Jose:Angular josefranciscumurciafuentes$ docker-compose down
WARN[0000] /Users/josefranciscumurciafuentes/Angular/compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion
[+] Running 4/4
✓ Container angular-postgres-nginx Removed 0.1s
✓ Container angular-app-1 Removed 0.1s
✓ Container angular-postgres-1.1 Removed 0.1s
✓ Network angular_net_jfmf Removed 0.1s
MacBook-Pro-de-Jose:Angular josefranciscumurciafuentes$
```