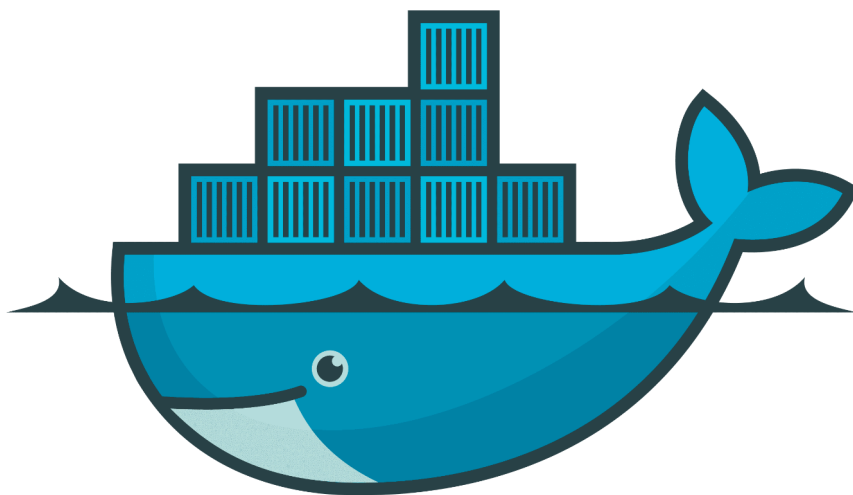


Práctica Docker

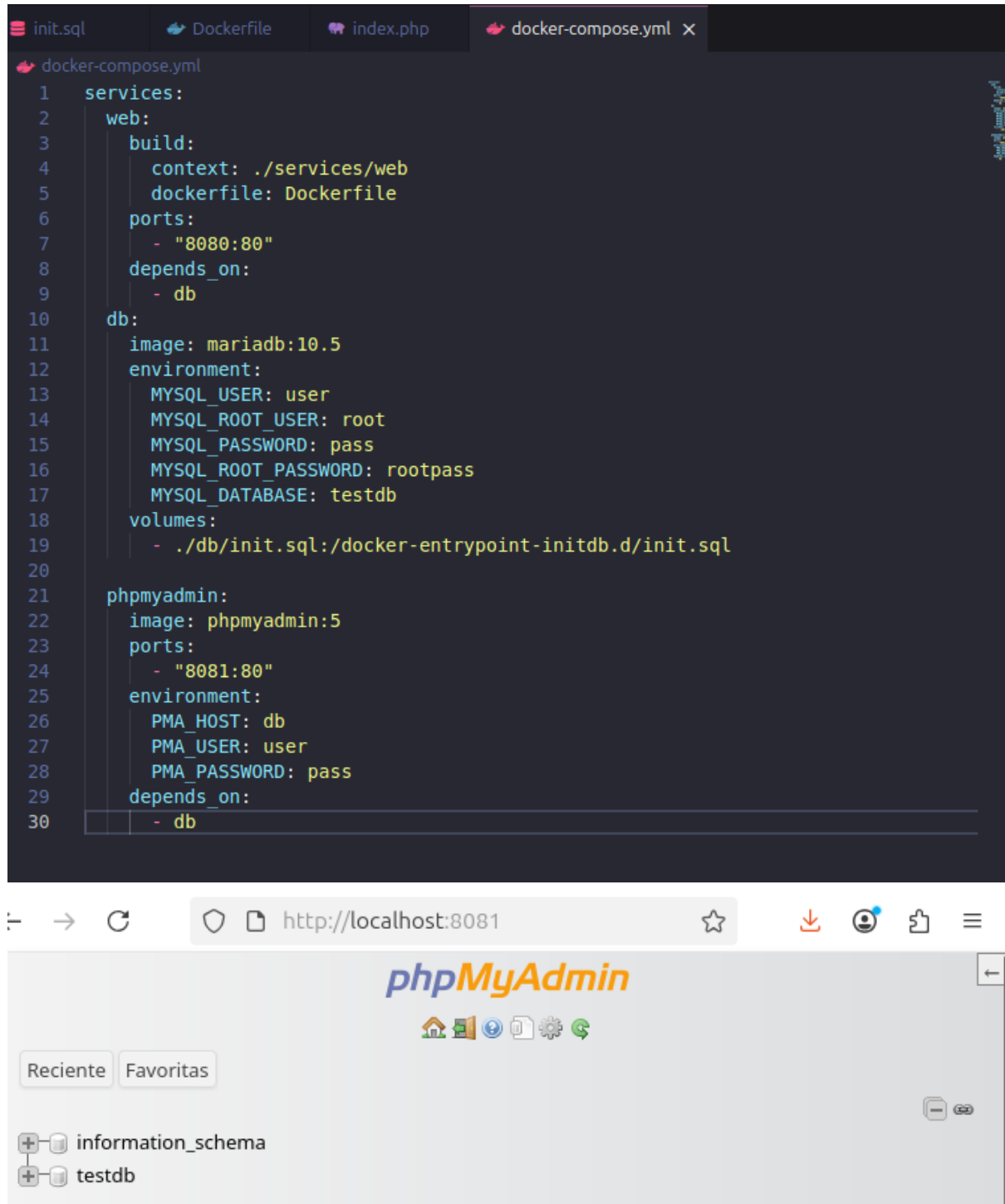


docker

Manuel Morente Díaz

Añadimos la interfaz a la BD

Se añade phpmyadmin en el docker-compose.yml para acceder a la web en cuanto a su interfaz y poder gestionar la base de datos.

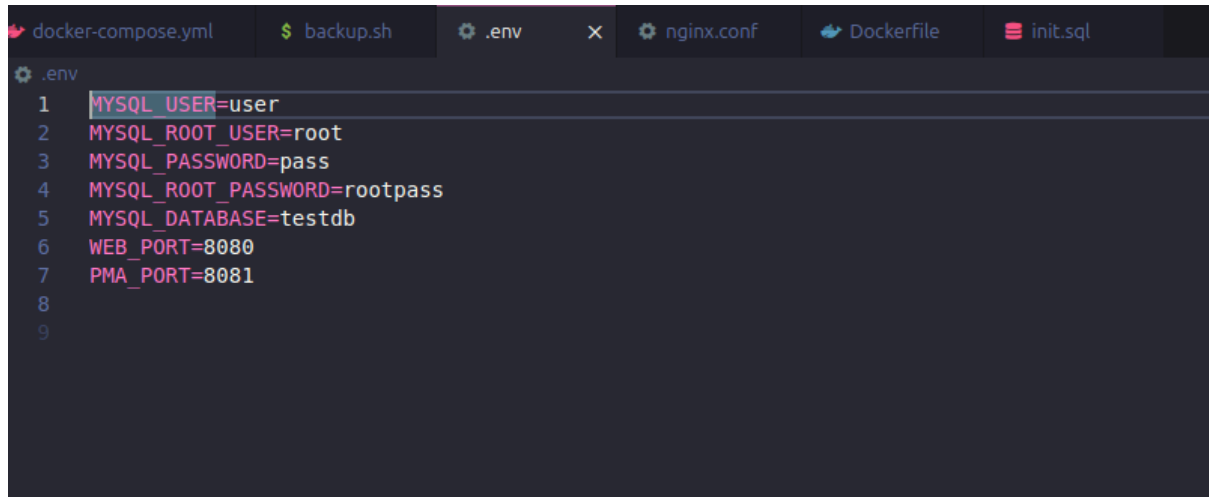


```
1 services:
2   web:
3     build:
4       context: ./services/web
5       dockerfile: Dockerfile
6     ports:
7       - "8080:80"
8     depends_on:
9       - db
10  db:
11    image: mariadb:10.5
12    environment:
13      MYSQL_USER: user
14      MYSQL_ROOT_USER: root
15      MYSQL_PASSWORD: pass
16      MYSQL_ROOT_PASSWORD: rootpass
17      MYSQL_DATABASE: testdb
18    volumes:
19      - ./db/init.sql:/docker-entrypoint-initdb.d/init.sql
20
21  phpmyadmin:
22    image: phpmyadmin:5
23    ports:
24      - "8081:80"
25    environment:
26      PMA_HOST: db
27      PMA_USER: user
28      PMA_PASSWORD: pass
29    depends_on:
30      - db
```

The browser window shows the phpMyAdmin interface at <http://localhost:8081>. The interface includes the phpMyAdmin logo, navigation icons, and tabs for 'Reciente' and 'Favoritas'. The left sidebar shows a tree view with 'information_schema' and 'testdb' databases.

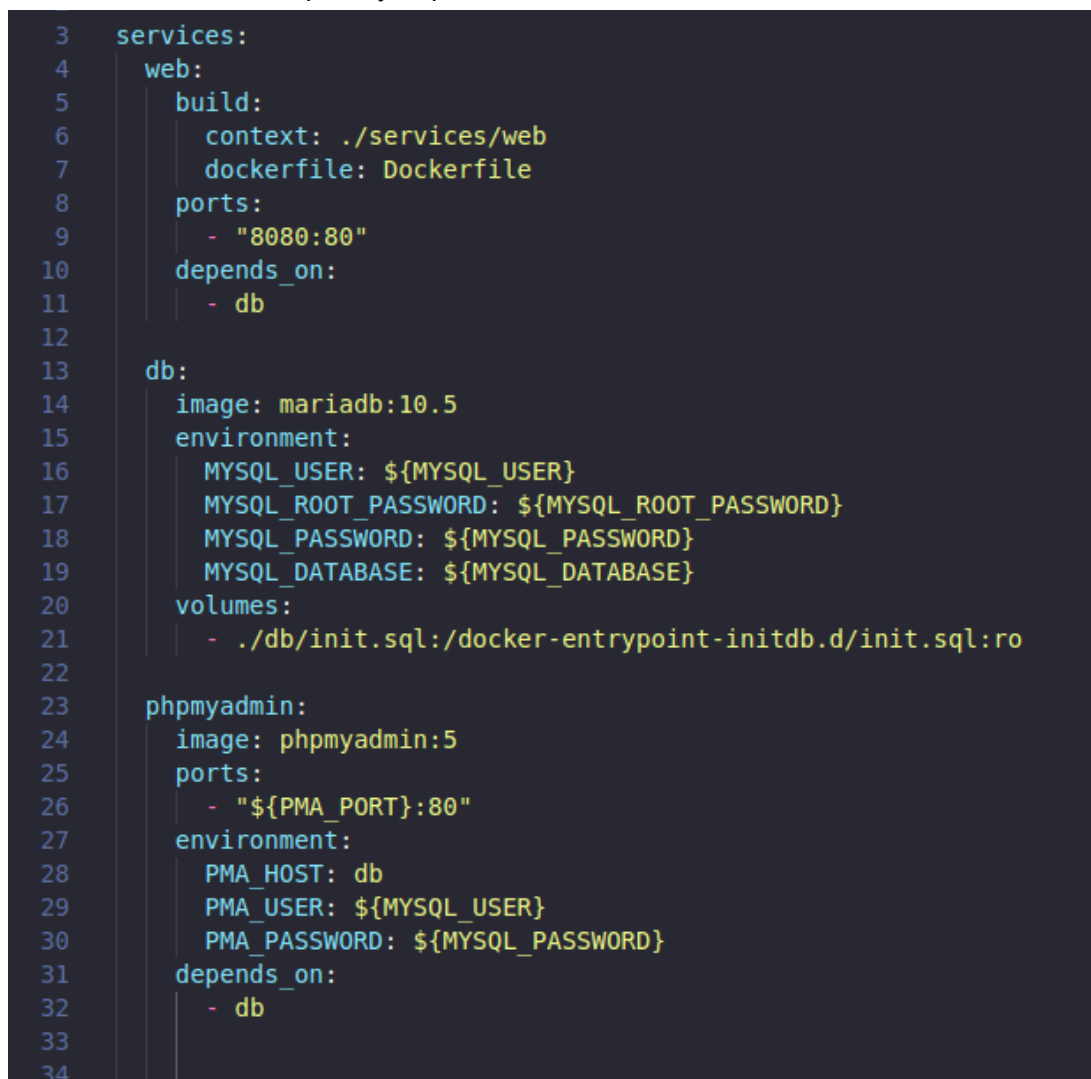
Añadimos variables de entorno en un .env

Creo el archivo .env que se encuentra en la carpeta raíz, en este se encuentran claves y entornos ya usados. (Es tan solo un .env, sin nombre ya que de lo contrario no lo reconoce como tal)

A screenshot of a code editor with several tabs at the top: 'docker-compose.yml', '\$ backup.sh', '.env', 'nginx.conf', 'Dockerfile', and 'init.sql'. The '.env' tab is active, showing a list of environment variables. The variables are: 'MYSQL_USER=user', 'MYSQL_ROOT_USER=root', 'MYSQL_PASSWORD=pass', 'MYSQL_ROOT_PASSWORD=rootpass', 'MYSQL_DATABASE=testdb', 'WEB_PORT=8080', and 'PMA_PORT=8081'. The lines are numbered from 1 to 9.

```
.env
1 MYSQL_USER=user
2 MYSQL_ROOT_USER=root
3 MYSQL_PASSWORD=pass
4 MYSQL_ROOT_PASSWORD=rootpass
5 MYSQL_DATABASE=testdb
6 WEB_PORT=8080
7 PMA_PORT=8081
8
9
```

Modifico el docker-compose.yml para usar las variables del .env.

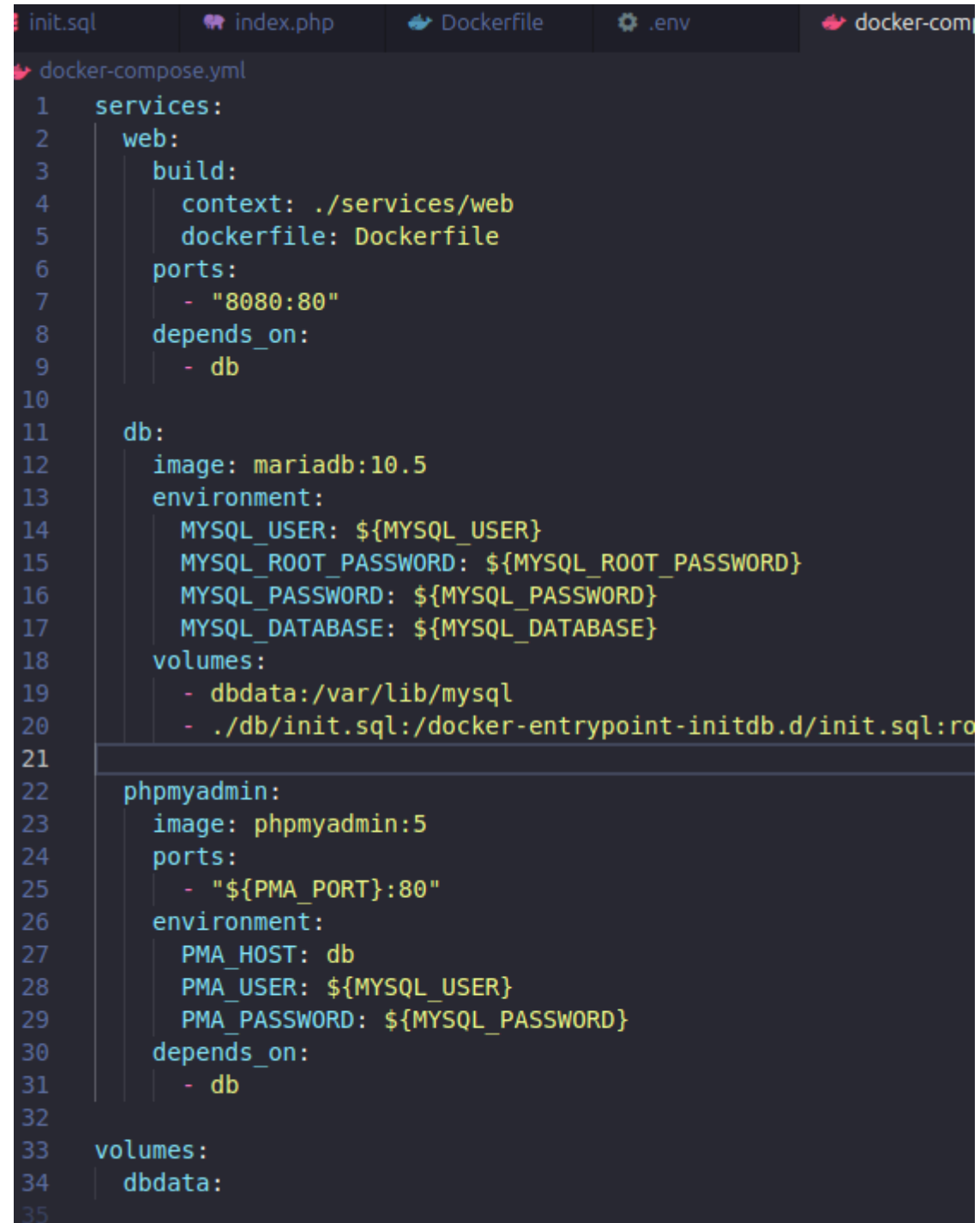
A screenshot of a code editor showing the 'docker-compose.yml' file. The file defines three services: 'web', 'db', and 'phpmyadmin'. The 'web' service uses a 'Dockerfile' context and depends on the 'db' service. The 'db' service uses the 'mariadb:10.5' image and its environment is configured with variables from the '.env' file. The 'phpmyadmin' service uses the 'phpmyadmin:5' image and its environment is also configured with variables from the '.env' file. The lines are numbered from 3 to 34.

```
3 services:
4   web:
5     build:
6       context: ./services/web
7       dockerfile: Dockerfile
8     ports:
9       - "8080:80"
10    depends_on:
11      - db
12
13   db:
14     image: mariadb:10.5
15     environment:
16       MYSQL_USER: ${MYSQL_USER}
17       MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
18       MYSQL_PASSWORD: ${MYSQL_PASSWORD}
19       MYSQL_DATABASE: ${MYSQL_DATABASE}
20     volumes:
21       - ./db/init.sql:/docker-entrypoint-initdb.d/init.sql:ro
22
23   phpmyadmin:
24     image: phpmyadmin:5
25     ports:
26       - "${PMA_PORT}:80"
27     environment:
28       PMA_HOST: db
29       PMA_USER: ${MYSQL_USER}
30       PMA_PASSWORD: ${MYSQL_PASSWORD}
31     depends_on:
32       - db
33
34
```

Añadimos persistencia al contenedor

Dentro de db, añado un nuevo volumen, el cual es dbdata:/var/lib/mysql, indicando la ruta de guardado.

Debajo de todo el archivo, al final de este en volumes, se llama el volumen creado, dígame dbdata



```
init.sql  index.php  Dockerfile  .env  docker-com
docker-compose.yml
1  services:
2    web:
3      build:
4        context: ./services/web
5        dockerfile: Dockerfile
6      ports:
7        - "8080:80"
8      depends_on:
9        - db
10
11    db:
12      image: mariadb:10.5
13      environment:
14        MYSQL_USER: ${MYSQL_USER}
15        MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
16        MYSQL_PASSWORD: ${MYSQL_PASSWORD}
17        MYSQL_DATABASE: ${MYSQL_DATABASE}
18      volumes:
19        - dbdata:/var/lib/mysql
20        - ./db/init.sql:/docker-entrypoint-initdb.d/init.sql:ro
21
22    phpmyadmin:
23      image: phpmyadmin:5
24      ports:
25        - "${PMA_PORT}:80"
26      environment:
27        PMA_HOST: db
28        PMA_USER: ${MYSQL_USER}
29        PMA_PASSWORD: ${MYSQL_PASSWORD}
30      depends_on:
31        - db
32
33  volumes:
34    dbdata:
```

Distinguimos entre servicios

Creo en network las redes backend y frontend, db en este caso está solo en el backend y phpmyadmin requiere de ambos, e indico ambas redes al final de todo.

```
init.sql  index.php  Dockerfile  .env  docker-compose.yml
❯ docker-compose.yml
1  services:
2    web:
3      build:
4        context: ./services/web
5        dockerfile: Dockerfile
6      ports:
7        - "${WEB_PORT}:80"
8      depends_on:
9        - db
10
11     networks: [frontend, backend]
12
13   db:
14     image: mariadb:10.5
15     environment:
16       MYSQL_USER: ${MYSQL_USER}
17       MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
18       MYSQL_PASSWORD: ${MYSQL_PASSWORD}
19       MYSQL_DATABASE: ${MYSQL_DATABASE}
20     volumes:
21       - dbdata:/var/lib/mysql
22       - ./db/init.sql:/docker-entrypoint-initdb.d/init.sql:ro
23     networks: [backend]
24
25   phpmyadmin:
26     image: phpmyadmin:5
27     ports:
28       - "${PMA_PORT}:80"
29     environment:
30       PMA_HOST: db
31       PMA_USER: ${MYSQL_USER}
32       PMA_PASSWORD: ${MYSQL_PASSWORD}
33     depends_on:
34       - db
35     networks: [frontend, backend]
36
37   volumes:
38     dbdata:
39
40   networks:
41     frontend:
42     backend:
```

Check de la base de datos

Con healthcheck, dentro de la db, ejecutamos mysqladmin ping, indicando este mismo. En el caso de la web, se define una condición en el que service_healthy esté disponible antes de arrancar la db.

```

1 services:
2   web:
3     image: phpmyadmin/phpmyadmin
4     ports:
5       - "${WEB_PORT}:80"
6     depends_on:
7       db:
8         condition: service_healthy
9     networks: [frontend, backend]
10
11   db:
12     image: mariadb:10.5
13     environment:
14       MYSQL_USER: ${MYSQL_USER}
15       MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
16       MYSQL_PASSWORD: ${MYSQL_PASSWORD}
17       MYSQL_DATABASE: ${MYSQL_DATABASE}
18     volumes:
19       - dbdata:/var/lib/mysql
20       - ./db/init.sql:/docker-entrypoint-initdb.d/init.sql:ro
21     networks: [backend]
22
23     healthcheck:
24       test: ["CMD-SHELL", "mysqladmin ping -h 127.0.0.1 -uroot -p`${MYSQL_ROOT_PASSWORD}` --silent"]
25       interval: 5s
26       timeout: 3s
27       retries: 20
28       start_period: 20s
29
30   phpmyadmin:
31     image: phpmyadmin/phpmyadmin
32     ports:
33       - "${PMA_PORT}:80"
34     environment:
35       PMA_HOST: db
36       PMA_USER: ${MYSQL_USER}
37       PMA_PASSWORD: ${MYSQL_PASSWORD}
38     depends_on:
39       db:
40         condition: service_healthy
41     networks: [frontend, backend]
42
43 volumes:
44   dbdata:
45
46 networks:
47   frontend:
48   backend:

```

Certificado autofirmado

Con este comando genero el certificado, creando los archivos dentro de la carpeta certs, es decir, selfsigned.key y selfsigned.crt.

[illegible]

En el apartado web dentro de docker-compose, cambio los puertos por expose, dejando el 80, en el proxy, defino el archivo y carpeta nginx, que sirve TLS con los certificados y hacer reverse proxy hacia la web.

```
docker-compose.yml
1  services:
2    web:
3      build:
4        context: ./services/web
5        dockerfile: Dockerfile
6      expose: ["80"]
7      ports:
8        - "${WEB_PORT}:80"
9      depends_on:
10        db:
11          condition: service_healthy
12
13      networks: [frontend, backend]
14
15 proxy:
16   image: nginx:alpine
17   ports:
18     - "88:80"
19     - "8443:443"
20   volumes:
21     - ./nginx.conf:/etc/nginx/conf.d:r
22     - ./certs:/etc/nginx/certs:r
23   depends_on:
24     web:
25       condition: service_started
```

Este es el nuevo archivo creado mediante el proxy anterior, en este lo que hace es redirigir HTTP y HTTPS y el proxy_pass a web, que es el nombre del servicio en Docker Compose

```
Dockerfile  .env  docker-compose.yml  nginx.conf  init.sql  index.php
services > web > nginx.conf
1  server {
2    listen 80;
3    server_name localhost;
4    return 301 https://$host:8443$request_uri;
5  }
6
7  server {
8    listen 443 ssl;
9    server_name localhost;
10
11    ssl_certificate      /etc/nginx/certs/selfsigned.crt;
12    ssl_certificate_key  /etc/nginx/certs/selfsigned.key;
13
14    set $app web;
15
16    location / {
17      proxy_pass http://$app:80;
18      proxy_set_header Host      $host;
19      proxy_set_header X-Real-IP  $remote_addr;
20      proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
21      proxy_set_header X-Forwarded-Proto $scheme;
22      proxy_http_version 1.1;
23      proxy_set_header Connection "";
24    }
25  }
26
```

Balanceo de carga

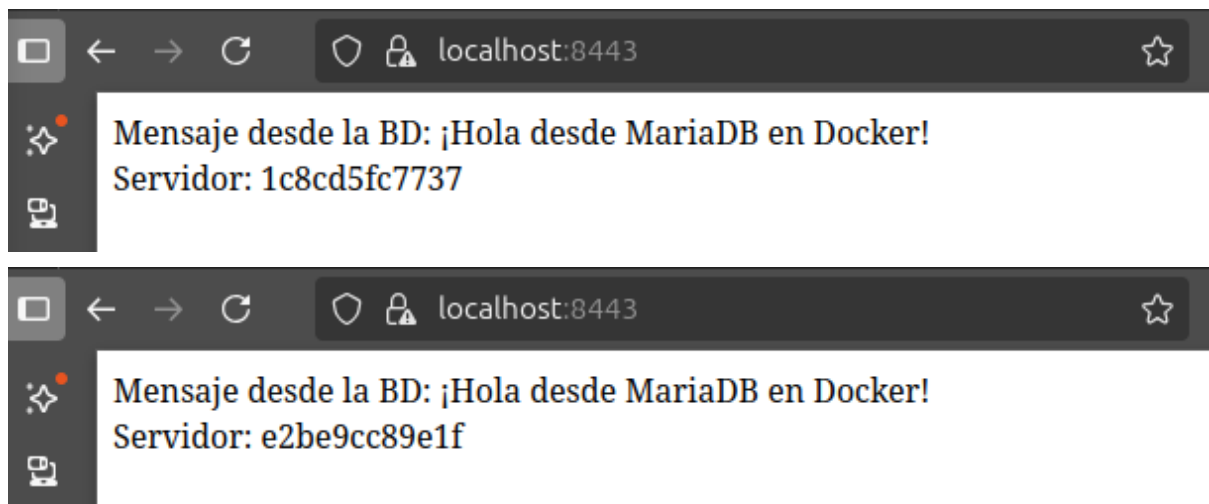
Desde el archivo index.php, añado al final la línea de código que muestra el nombre del host o hostname del contenedor, luego con el comando correspondiente lo lanzo:

docker compose up -d --scale web=2



```
services > web > src > index.php
1  <?php
2  $conn = new mysqli("db", "user", "pass", "testdb");
3  if ($conn->connect_error) {
4      die("Error de conexión: " . $conn->connect_error);
5  }
6  $result = $conn->query("SELECT mensaje FROM mensajes");
7  $row = $result->fetch_assoc();
8  echo "Mensaje desde la BD: " . $row["mensaje"] . "<br>";
9
10 echo "Servidor: " . gethostname();
11
12
```

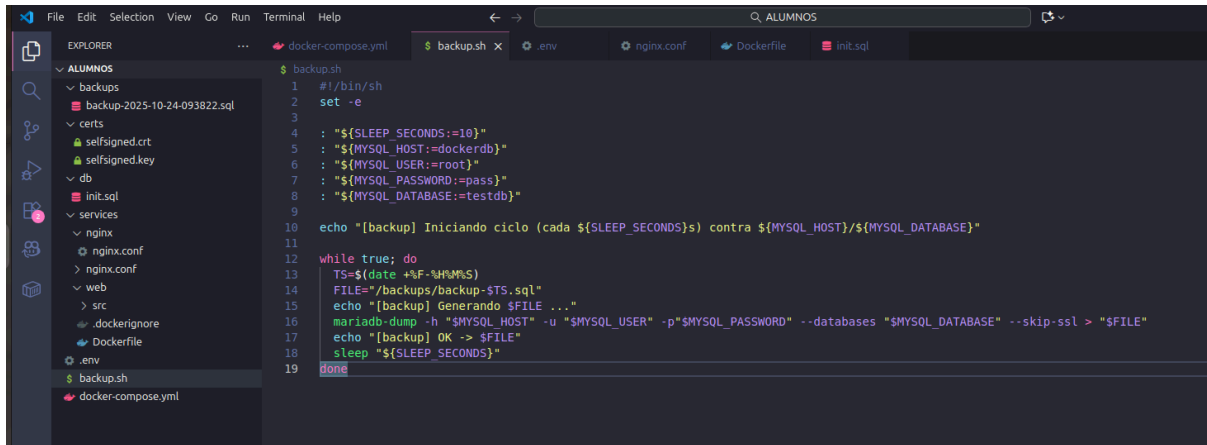
Cada que actualice la página, la web muestra distintos caracteres, indicando su correcto funcionamiento.



Backups

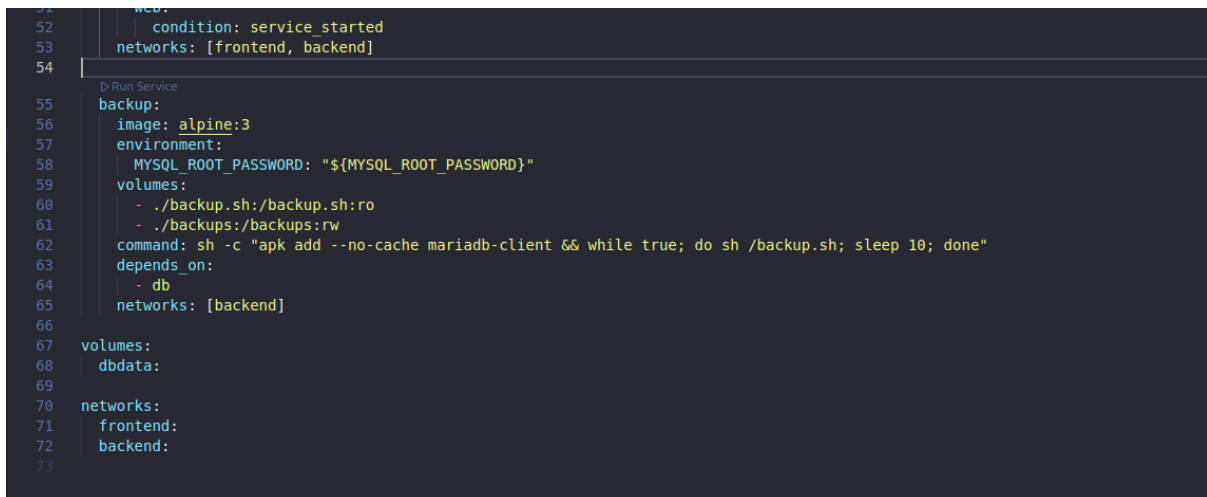
Se crea un nuevo archivo, [backup.sh](#), aunque cambiando la cantidad de tiempo en la que cada vez hace el backup, de 60 a 10 segundos.

Esto es un script que cada que pase ese x tiempo, guarda este mismo backup en donde se le indique.



```
1 #!/bin/sh
2 set -e
3
4 : "${SLEEP_SECONDS:=10}"
5 : "${MYSQL_HOST:=dockerdb}"
6 : "${MYSQL_USER:=root}"
7 : "${MYSQL_PASSWORD:=pass}"
8 : "${MYSQL_DATABASE:=testdb}"
9
10 echo "[backup] Iniciando ciclo (cada ${SLEEP_SECONDS}s) contra ${MYSQL_HOST}/${MYSQL_DATABASE}"
11
12 while true; do
13     TS=$(date +%F-%H%M%S)
14     FILE="/backups/backup-$TS.sql"
15     echo "[backup] Generando $FILE ..."
16     mariadb-dump -h "$MYSQL_HOST" -u "$MYSQL_USER" -p"$MYSQL_PASSWORD" --databases "$MYSQL_DATABASE" --skip-ssl > "$FILE"
17     echo "[backup] OK -> $FILE"
18     sleep "$SLEEP_SECONDS"
19 done
```

En docker compose, este fragmento monta la carpeta backups para guardar los anteriormente nombrados, y ejecuta el archivo backup.sh



```
52 condition: service_started
53 networks: [frontend, backend]
54
55 backup:
56   image: alpine:3
57   environment:
58     MYSQL_ROOT_PASSWORD: "${MYSQL_ROOT_PASSWORD}"
59   volumes:
60     - ./backup.sh:/backup.sh:ro
61     - ./backups:/backups:rw
62   command: sh -c "apk add --no-cache mariadb-client && while true; do sh /backup.sh; sleep 10; done"
63   depends_on:
64     - db
65   networks: [backend]
66
67 volumes:
68   dbdata:
69
70 networks:
71   frontend:
72   backend:
```

Servicio de aplicaciones

Tras descargar sample.war, dentro de la ruta services/tomcat/webapps/sample.war, en el docker compose, añado el servicio tomcat, que expone el puerto 8080 en la red Docker, se monta el sample war en webapps para que este se despliegue.

```
Run Service
tomcat:
  image: tomcat:9.0
  expose: ["8080"]
  volumes:
    - ./services/tomcat/webapps/sample.war:/usr/local/tomcat/webapps/sample.war
  networks: [backend]

volumes:
  dbdata:

networks:
  frontend:
  backend:
```

Proxy configurado para enrutar sample, a tomcat:8080/sample/ en archivo nginx.conf

```
}

server{
    listen 8080;
    server_name localhost;

    location /sample/ {
        proxy_pass http://tomcat:8080/sample/;
        proxy_set_header Host $host;
    }
}
```

Resultado final tras ejecución:

