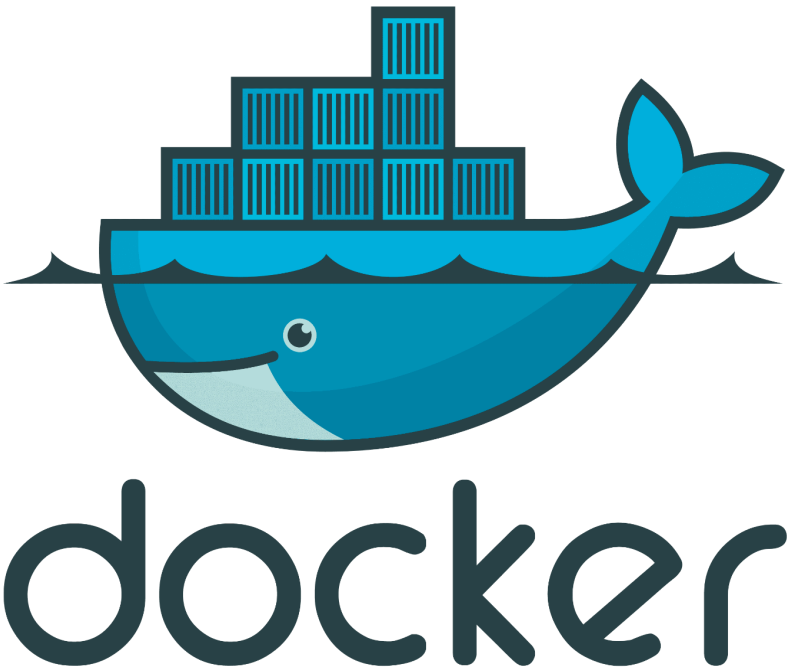
Pràctica 6:
PHP+MariaDB emprant dockers



Guillem Serrat Bahí

Index

Introducció

En aquesta pràctica treballarem amb Dockers per posar en producció diverses aplicacions web.

Realitzarem una instal·lació de Docker sobre un entorn Debian 13

Seguidament treballarem amb Dockerfiles per crear imatges de Dockers i crearem un contenidor senzill a partir d'aquesta imatge

A continuació treballarem amb Docker Compose, per posar en marxa diversos Dockers simultàneament en la mateixa xarxa, connectant serveis PHP i MariaDB

Per últim, migrarem tot l'entorn de Docker a un VPS, on crearem dominis per accés públic i certificarem aquests dominis per fer servir HTTPS

VPS

Tant aquesta pràctica com la resta d'aquestes i altres projectes es poden consultar dins de la pàgina web www.gserrat.cat.

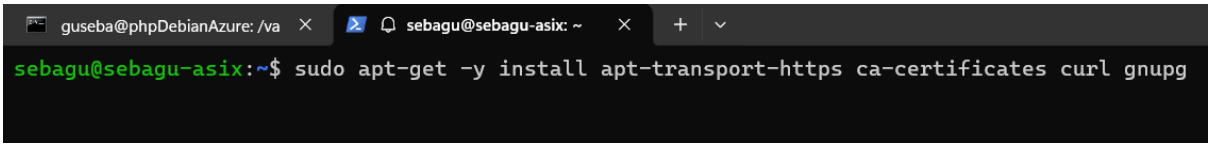
A més a més, a la pàgina web www.wiki.gserrat.cat es troba una wiki sobre la pràctica realitzada.

Enllaços ràpids:

- Pàgina inicial de la pràctica 6.2
- Pàgina info.php de la pràctica 6.3
- Pàgina 00_connect de la pràctica 6.4
- Pàgina inicial del projecte de la pràctica 6.5
- Pàgina de PhpMyAdmin del projecte de la pràctica 6.5

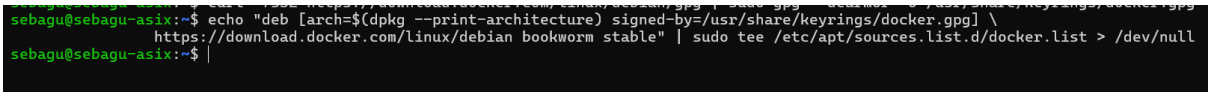
Pràctica 6.1

Primerament, per poder treballar amb Dockers caldrà instal·lar els paquets requerits:
sudo apt-get -y install apt-transport-https ca-certificates curl gnupg



A continuació, caldrà descarregar les claus GPG del repositori Docker, les quals serveixen per verificar l'autenticitat dels paquets Docker quan s'instal·lin o s'actualitzain

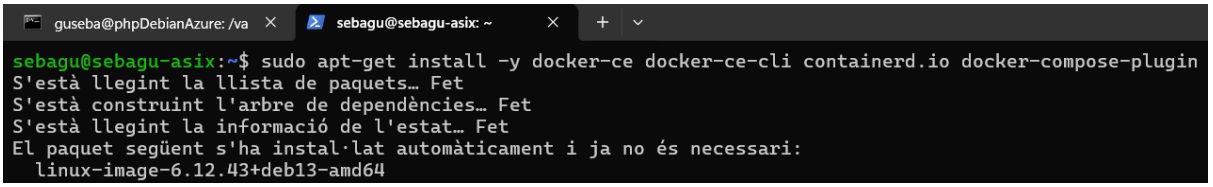
A continuació afegirem a la llista de repositoris el repositori oficial de Docker per poder instal·lar les seves utilitats



Un cop actualitzada la llista de repositoris, haurem d'actualitzar la llista de paquets disponibles tenint en compte el nou repositori de docker

Seguidament, instal·larem totes les utilitats de Docker:

- docker-ce, per poder treballar amb Dockers
- docker-ce-cli, per poder treballar amb Dockers des de la terminal
- containerd.io, motor utilitzat internament per Docker
- docker-compose-plugin, per poder treballar amb Docker compose



Un cop instal·lat, podem comprovar la seva versió amb l'ordre *docker version*

```
sebagu@sebagu-asix:~$ docker version
Client: Docker Engine - Community
 Version:           29.1.4
 API version:       1.52
 Go version:        go1.25.5
 Git commit:        0e6fee6
 Built:             Thu Jan  8 19:56:47 2026
 OS/Arch:           linux/amd64
 Context:           default
permission denied while trying to connect to the docker API at unix:///var/run/docker.sock
sebagu@sebagu-asix:~$
```

I per comprovar que el Docker compose s'ha instal·lat correctament, ho farem amb l'ordre *docker compose version*

```
sebagu@sebagu-asix:~$ docker compose version
Docker Compose version v5.0.1
sebagu@sebagu-asix:~$
```

Per últim, per tal de que el nostre usuari pugui treballar amb Dockers sense necessitat de privilegis elevats, afegirem el nostre usuari al grup docker

```
guseba@phpDebianAzure: /va   sebagu@sebagu-asix: ~
sebagu@sebagu-asix:~$ sudo gpasswd -a sebagu docker
```

Pràctica 6.2

Primerament, crearem un directori que serà el nostre entorn de treball

```
sebagu@sebagu-asix:/var/www/html/php/exercicis/practica6/6.2$ mkdir php-docker-app
sebagu@sebagu-asix:/var/www/html/php/exercicis/practica6/6.2$ ls
php-docker-app
sebagu@sebagu-asix:/var/www/html/php/exercicis/practica6/6.2$ cd php-docker-app/
sebagu@sebagu-asix:/var/www/html/php/exercicis/practica6/6.2/php-docker-app$
```

Seguidament crearem un fitxer index.html i introduirem el codi de la pràctica 2.11. Degut a que el nom del fitxer és diferent que el que era quan es va realitzar la pràctica, haurem de canviar:

- L'acció del formulari a "index.php"
- El "show source" del codi a "index.php

```
sebagu@sebagu-asix: /var/ww
 GNU nano 8.4                                    index.php *
<!DOCTYPE html>
<html lang="ca">
    <head>
        <title>Exercici 11 Guillem Serrat</title>
        <meta http-equiv="Content-type" content="text/html; charset=UTF-8">
    </head>
    <body>
        <form method="POST" action="index.php">
            <p>Nombre de files a realitzar: <input type="text" name="files" required/></p>
            <p>Símbol: <input type="text" name="simbol" required/></p>
            <p><button type="submit">Calcula</button></p>
        </form>

        <?php
            $files = $_POST["files"];
            $espai = " ";              // Espai buit
            $caracter = $espai . $_POST["simbol"];    // Caràcter per a la piràmide

            if (!empty($files)) {
                for ($i = 1; $i <= $files; $i++) { // Comença el bucle per tantes files s'hagin especificat
                    for ($j = $i; $j < $files; $j++) { // Imprimeix els espais inicials (Línia actual (j = i) fins a línias -1 ())
                        echo $espai;
                    }
                    for ($k = 1; $k <= $i; $k++) { // Imprimeix els símbols. Si la línia (i) és 1, imprimeix 1 símbol. Tants símbols com número de línia
                        echo $caracter;
                    }
                    echo "<br>\n"; // Nova línia
                }
            }
        ?>
        <h1>Codi en PHP</h1>
        <?php
        show_source("index.php");
        ?>
    </body>
</html>
```

A continuació crearem un Dockerfile. Un Dockerfile conté les instruccions necessàries per construir una imatge de Docker.

Dins d'un Dockerfile hem de definir com a mínim una imatge base. Aquesta imatge pot ser un sistema operatiu (Debian, Ubuntu) o un programari en concret (Apache, Nginx, MariaDB, etc)

En aquest cas definim que la imatge és php:7.0-apache, és a dir, un servidor Apache preparat per treballar amb PHP

A més a més, amb la instrucció COPY estem indicant que tot el que hi ha en aquest directori (el fitxer index.php) es copiï al directori /var/www/html del contenidor Docker

```
sebagu@sebagu-asix:/var/www/html/php/exercicis/practica6/6.2/php-docker-app$ nano Dockerfile
sebagu@sebagu-asix:/var/www/html/php/exercicis/practica6/6.2/php-docker-app$ cat Dockerfile
FROM php:7.0-apache
COPY . /var/www/html
sebagu@sebagu-asix:/var/www/html/php/exercicis/practica6/6.2/php-docker-app$
```

Un cop definit el Dockerfile, crearem la imatge amb l'ordre *docker build -t <nomImatge> .*

```
sebagu@sebagu-asix:/var/www/html/php/exercicis/practica6/6.2/php-docker-app$ docker build -t php-app .
[+] Building 47.2s (7/7) FINISHED                                                                    docker:default
 => [internal] load build definition from Dockerfile                                                         0.1s
 => => transferring dockerfile: 78B                                                                          0.0s
 => [internal] load metadata for docker.io/library/php:7.0-apache                                            2.4s
 => [internal] load .dockerignore                                                                            0.1s
 => => transferring context: 2B                                                                              0.0s
 => [internal] load build context                                                                            0.1s
```

Un cop hem creat la imatge, la podrem cercar amb l'ordre *docker imatges,* la qual mostrarà les imatges tan pròpies creades per un Dockerfile o públiques descarregades de Dockerhub

```
sebagu@sebagu-asix:/var/www/html/php/exercicis/practica6/6.2/php-docker-app$ docker images
                                                                                    i Info →  U In Use
IMAGE               ID             DISK USAGE   CONTENT SIZE   EXTRA
hello-world:latest  d4aaab6242e0       25.9kB         9.52kB    U
php-app:latest      0e13ed3185ff        529MB          133MB
sebagu@sebagu-asix:/var/www/html/php/exercicis/practica6/6.2/php-docker-app$
```

A partir d'aquesta imatge generarem el contenidor Docker amb l'ordre *docker run <nomImatge> &.* Fem servir & per executar-ho en segon pla

```
sebagu@sebagu-asix:/var/www/html/php/exercicis/practica6/6.2/php-docker-app$ docker run php-app &
[1] 49810
sebagu@sebagu-asix:/var/www/html/php/exercicis/practica6/6.2/php-docker-app$ AH00558: apache2: Could not reliably determine the server's fully qualified dom
ain name, using 172.17.0.2. Set the 'ServerName' directive globally to suppress this message
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 172.17.0.2. Set the 'ServerName' directive globally to suppre
ss this message
[Tue Jan 13 18:38:09.592567 2026] [mpm_prefork:notice] [pid 1] AH00163: Apache/2.4.25 (Debian) PHP/7.0.33 configured -- resuming normal operations
[Tue Jan 13 18:38:09.592850 2026] [core:notice] [pid 1] AH00094: Command line: 'apache2 -D FOREGROUND'
```

Amb l'ordre *docker ps -a* podem comprovar els contenidors creats i el seu estat. En aquest cas tenim el contenidor de prova "hello" i el contenidor que hem acabat de crear.

```
guseba@phpDebianAzure: /va ×   sebagu@sebagu-asix: /var/ww ×   +
sebagu@sebagu-asix:/var/www/html/php/exercicis/practica6/6.2/php-docker-app$ docker ps -a
CONTAINER ID   IMAGE         COMMAND                  CREATED          STATUS                      PORTS     NAMES
387e3bb1f709   php-app       "docker-php-entrypoi…"   About a minute ago   Up About a minute           80/tcp    determined_booth
daa52889be55   hello-world   "/hello"                 18 minutes ago       Exited (0) 18 minutes ago           focused_shockley
sebagu@sebagu-asix:/var/www/html/php/exercicis/practica6/6.2/php-docker-app$
```

Tot i això, si comprovem els ports exportats, veurem que no apareix cap port 80 i per tant l'aplicació no serà accesible. Això és degut que al Dockerfile no hem especificat que el contenidor exportarà aquest port

```
sebagu@sebagu-asix:/var/www/html/php/exercicis/practica6/6.2/php-docker-app$ sudo netstat -putan | grep 80
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN      1080/mariadbd
sebagu@sebagu-asix:/var/www/html/php/exercicis/practica6/6.2/php-docker-app$
```

Per exportar ports del Docker hem d'especifcar-ho al Dockerfile, amb la instrucció EXPOSE

```
sebagu@sebagu-asix:/var/www/html/php/exercicis/practica6/6.2/php-docker-app$ nano Dockerfile
sebagu@sebagu-asix:/var/www/html/php/exercicis/practica6/6.2/php-docker-app$ cat Dockerfile
FROM php:7.0-apache
COPY . /var/www/html
EXPOSE 80
sebagu@sebagu-asix:/var/www/html/php/exercicis/practica6/6.2/php-docker-app$
```

Degut a que hem modifcat el Dockerfile, la imatge generada anteriorment ja no és vàlida i n'hem de crear una de nova. En aquest cas especifcarem el mateix nom de la imatge però en versió 1.2

```
sebagu@sebagu-asix:/var/www/html/php/exercicis/practica6/6.2/php-docker-app$ docker build -t php-app:1.2
[+] Building 3.2s (7/7) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 88B
 => [internal] load metadata for docker.io/library/php:7.0-apache
 => [internal] load .dockerignore
```

Ara podem comprovar que tenim la 1.2 i la "latest". La latest no és realment la última versió, sent aquesta la 1.2, però ja que al crear la primera imatge no hem especificat cap versió, Docker la identifica com a "latest"

```
sebagu@sebagu-asix:/var/www/html/php/exercicis/practica6/6.2/php-docker-app$ docker images
                                                                                    Info →  U  In Use
IMAGE                ID             DISK USAGE   CONTENT SIZE   EXTRA
hello-world:latest   d4aaab6242e0       25.9kB         9.52kB    U
php-app:1.2          31a4afa90400c       529MB          133MB
php-app:latest       0e13ed3185ff        529MB          133MB    U
sebagu@sebagu-asix:/var/www/html/php/exercicis/practica6/6.2/php-docker-app$
```

Per generar el contenidor farem servir la mateixa ordre, afegint la flag *-p <portHost>:<portDocker>*

On "portHost" és el port que farà servir la màquina host al exportar el servei del Docker i "portDocker" el port intern del Docker

```
guseba@phpDebianAzure: /va ×   sebagu@sebagu-asix: /var/w ×   +
sebagu@sebagu-asix:/var/www/html/php/exercicis/practica6/6.2/php-docker-app$ docker run -d -p 80:80 php-app:1.2
039b94e0ffc6d914bcce5f65b12644894b5dd3fc79e17c216aa8d7c52e426bf5
sebagu@sebagu-asix:/var/www/html/php/exercicis/practica6/6.2/php-docker-app$
```

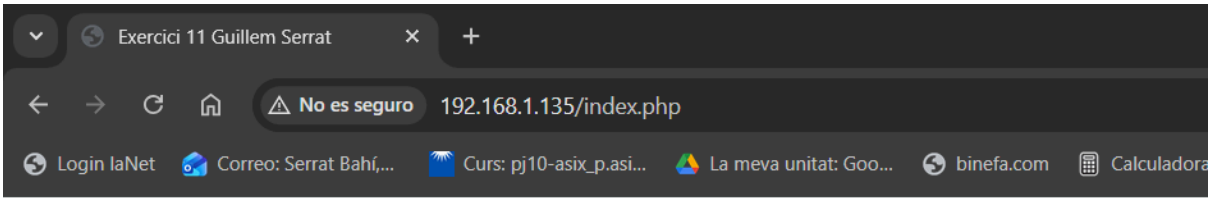
Si ara tornem a comprovar els ports, veurem que estem exportant el port 80 amb un servei anomenat “docker-proxy”

```
guseba@phpDebianAzure: /va   sebagu@sebagu-asix: /var/ww
sebagu@sebagu-asix:/var/www/html/php/exercicis/practica6/6.2/php-docker-app$ sudo netstat -putan | grep 80
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN      1080/mariadbd
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN      50750/docker-proxy
tcp6       0      0 :::80                   :::*                    LISTEN      50757/docker-proxy
sebagu@sebagu-asix:/var/www/html/php/exercicis/practica6/6.2/php-docker-app$
```

Seguidament només cal recuperar la IP de la màquina host

```
sebagu@sebagu-asix:/var/www/html/php/exercicis/practica6/6.2/php-docker-app$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:e1:77:99 brd ff:ff:ff:ff:ff:ff
    altname enx080027e17799
    inet 192.168.1.135/24 brd 192.168.1.255 scope global dynamic noprefixroute enp0s3
       valid_lft 78578sec preferred_lft 78578sec
    inet6 2a0c:5a86:f60c:3601::a4dc/128 scope global noprefixroute
       valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fee1:7799/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

I al entrar a la màquina host, veurem el codi PHP del Docker



Nombre de files a realitzar:

Símbol:

Calcula

```
  *
 * *
* * *
* * * *
* * * * *
```

Codi en PHP

```
<!DOCTYPE html>
<html lang="ca">
    <head>
        <title>Exercici 11 Guillem Serrat</title>
        <meta http-equiv="Content-type" content="text/html; charset=UTF-8">
    </head>
    <body>
        <form method="POST" action="index.php">
            <p>Nombre de files a realitzar: <input type="text" name="files" required/></p>
            <p>Símbol: <input type="text" name="simbol" required/></p>
            <p><button type="submit">Calcula</button></p>
```

Pràctica 6.3 i 6.4 (aprofitant el mateix compose)

Pràctica 6.3

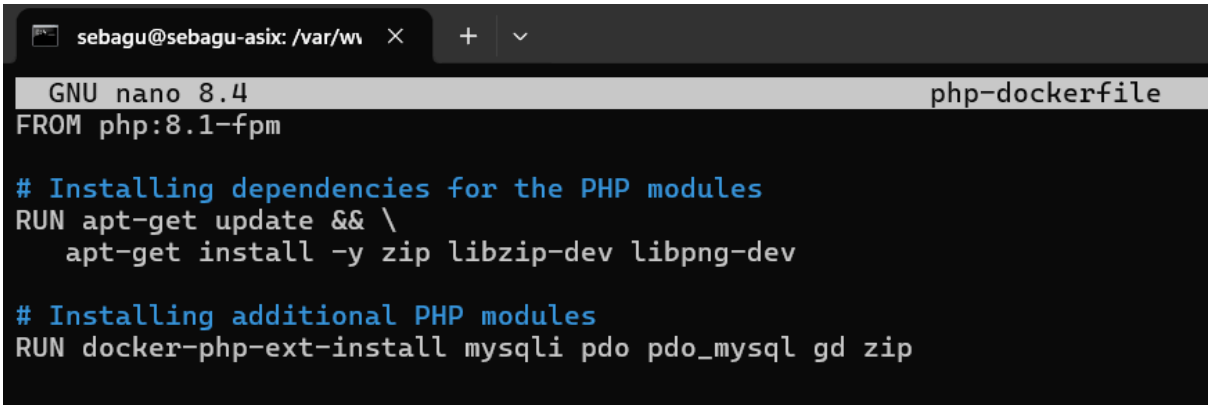
Quan treballem amb Dockers, sobretot quan necessitem una xarxa de Dockers conectada entre sí, fem servir el que s'anomena Docker Compose

Docker compose treballa amb fitxers docker-compose.yml, els quals descriuen:

- Serveis (o contenidors)
- Imatges d'aquests serveis
- Ports, xarxes, volums i variables d'entorn
- Dependències entre serveis

Per començar a treballar amb Docker Compose crearem la següent estructura de directoris

Seguidament, crearem el Dockerfile per un dels contenidors



Seguidament crearem un docker-compose.yml, on definirem:

1. Un contenidor anomenat PHP
    a. La imatge del contenidor serà a partir del Dockerfile (el propi compose fa el build)
    b. Un volum compartit entre la carpeta host ./php-files i /var/www/html
        i. Un volum compartit té la funció de mantenir els mateixos arxius en els directoris especificats entre el host i el docker
    c. Dependència del contenidor “mariadb”
        i. Aquest contenidor no es posarà en marxa fins que el contenidor mariadb no ho estigui
2. Un contenidor anomenat Nginx
    a. La imatge del contenidor serà la imatge oficial de Nginx, de dockerhub
    b. Els ports exposats seran
        i. El port 8090 de la màquina host dirigirà el port 80 del Docker
    c. Links (tot i que avui en dia no cal especificar-ho, ja que el propi compose crea una xarxa interna
        i. Un link al contenidor “php”, per permetre que tots dos contenidors es puguin comunicar
    d. Volums
        i. El contingut de ./php-files del host es compartirà amb el directori /var/www/html del Docker
        ii. El contingut de ./nginx-conf del host es compartirà amb el directori /etc/nginx/conf.d del Docker
    e. Dependència del contenidor amb nom “php”
        i. Aquest contenidor no es posarà en marxa fins que el contenidor php no ho estigui
3. Un contenidor anomenat mariadb (contenidor que únicament serà accesible des de la xarxa del docker compose i no des de l’exterior, i per això no cal exposar cap port)
    a. La imatge del contenidor serà la imatge oficial de MariaDB, de dockerhub
    b. Variables d’entorn
        i. La contrasenya de ROOT de MariaDB serà fjeclot
    c. Volums
        i. El contingut de /var/lib/mysql es desarà a un volum intern de Docker anomenat mysqldata (no es un directori del projecte, és un volum intern de docker, els quals s'acustuma a desar-se a /var/lib/docker/volumes)

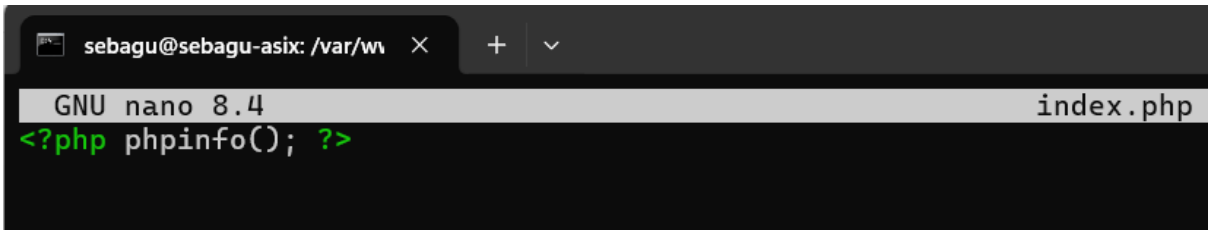
4. Un contenidor anomenat phpmyadmin
    a. La imatge del contenidor serà la imatge oficial de phpmyadmin, de dockerhub
    b. Variables d'entorn
        i. El hostname de la BD serà el nom del contenidor mariadb (Docker compose pot resoldre hostnames amb el nom del propi contenidor)
    c. Dependència del contenidor "mariadb"
        i. Aquest contenidor no es posarà en marxa fins que el contenidor mariadb no ho estigui

```
# version: '3.8' # L'atribut version ja no és necessari en l'última versió de docker

services:

  php:
    build:
      dockerfile: php-dockerfile #Construim el Docker a partir de la imatge pròpia
    volumes:
      - './php-files:/var/www/html' # Definim un volum. Els documents del directori ./php-files del host es compartiran amb el directori /var/www/html del Docker
    depends_on:
      - mariadb # Requerim que el servei mariadb estigui operatiu

  nginx:
    image: nginx:latest # Utilitzem la imatge oficial de nginx, la última versió
    ports:
      - 8090:80 # El port 8090 del host dirigirà al port 80 del Docker
    links: # Avui en dia no cal especificar-ho, ja que docker compose ja crea una xarxa interna
      - 'php' # Permet que el Docker Nginx es pugui comunicar amb el Docker "php"
    volumes:
      - './php-files:/var/www/html' # El contingut de ./php-files del host es compartirà amb el directori /var/www/html del Docker
      - './nginx-conf:/etc/nginx/conf.d' # El contingut de ./nginx-conf del host es compartirà amb el directori /etc/nginx/conf.d del Docker
    depends_on:
      - php # Requerim que el servei php estigui operatiu

  mariadb: # No exportem cap port ja que la BD no cal que sigui accesible des de l'exterior, sinó que únicament des dels altres Dockers
    image: mariadb:10.9 # Fem servir la imatge oficial de mariadb
    environment:
      MYSQL_ROOT_PASSWORD: fjeclot # Definim la contrasenya de root de MariaDB
    volumes:
      - mysqldata:/var/lib/mysql # El contingut de /var/lib/mysql es desarà a un volum intern de Docker anomenat mysqldata (NO ÉS UN DIRECTORI DEL PROJECTE, ÉS UN VOLUM INTERN DE DOCKER)

  phpmyadmin:
    image: phpmyadmin/phpmyadmin:latest # Fem servir la imatge de PHPMyAdmin
    ports:
      - 8091:80 # El port 8091 del host dirigirà al port 80 del Docker
    environment:
      PMA_HOST: mariadb # Indiquem el hostname de la BD, en aquest cas el nom del docker
    depends_on:
      - mariadb # Requerim que el servei mariadb estigui operatiu

# Definim els volums INTERNS de Docker
volumes:
  mysqldata: # Els arxius dels volumns interns de Docker s'acustuma a desar-se a /var/lib/docker/volumes
```

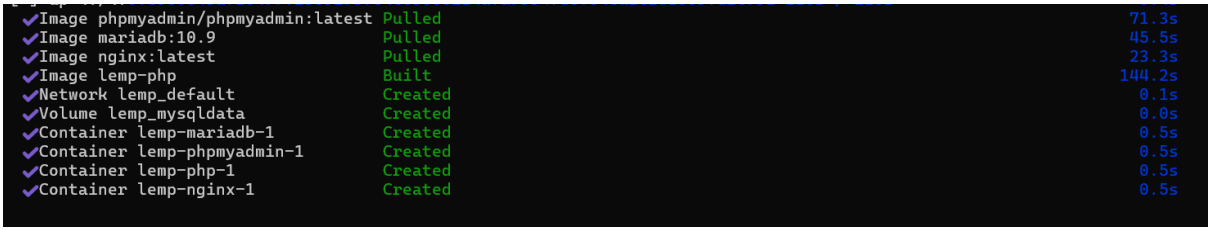
A continuació especificarem la següent configuració d'nginx

```
  GNU nano 8.4                                    nginx.conf *
server {
    listen 80 default_server; # Escoltem pel port 80, tot i que després el port 80 del Docker es converteix en el 8090 d>
    listen [::]:80 default_server;

    server_name localhost;

    root /var/www/html;
    index index.php index.html;

    location / {
        try_files $uri $uri/ /index.php?$args;
    }

    location ~* \.php$ {
        fastcgi_pass php:9000;
        include fastcgi_params;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        fastcgi_param SCRIPT_NAME $fastcgi_script_name;
    }
}
```

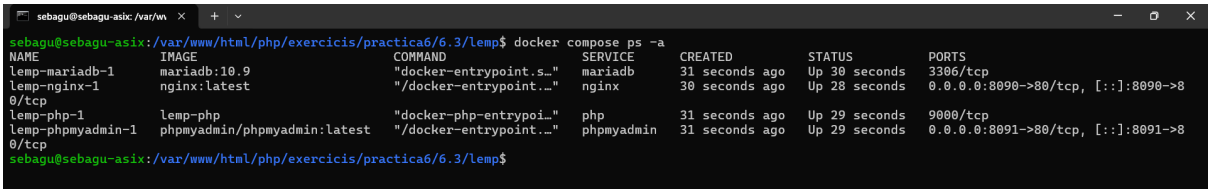
I per últim modificarem el fitxer index.php



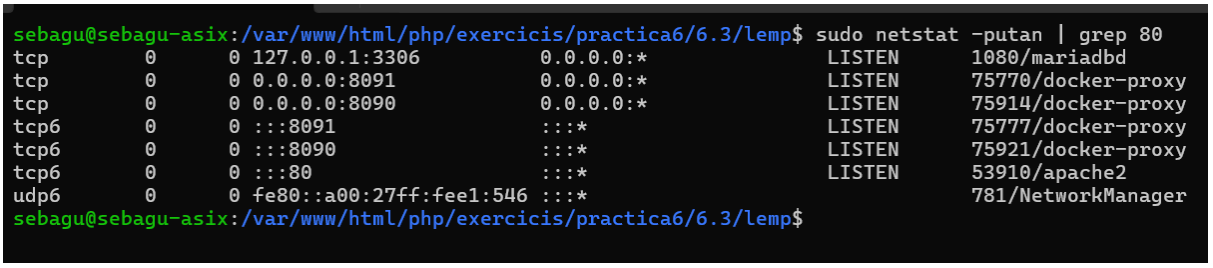
Un cop hem realitzat totes les configuracions necessàries, posarem en marxa tots els contenidors del docker compose amb l'ordre *docker compose up -d*



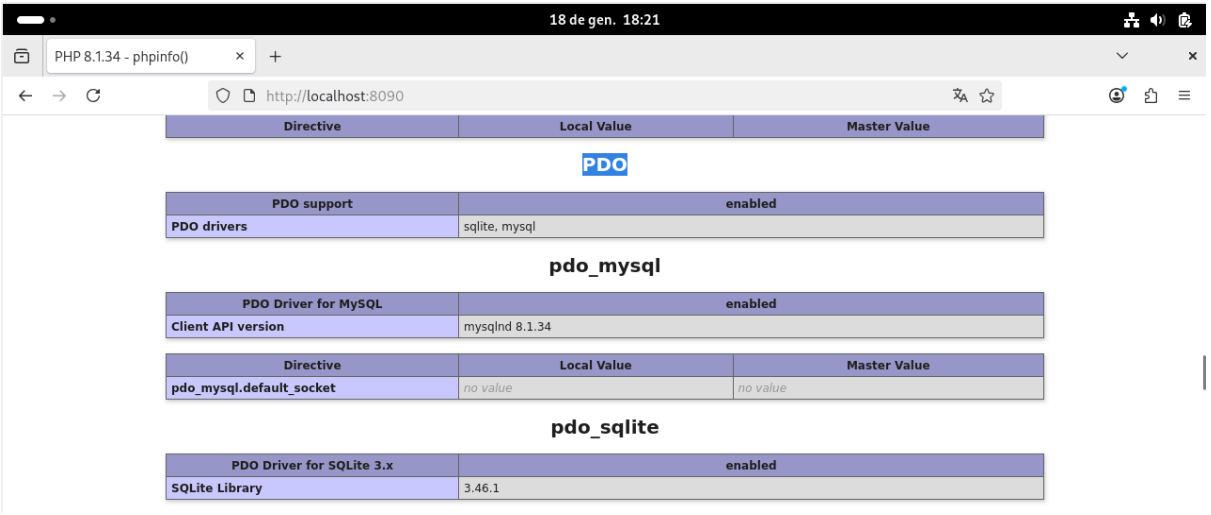
Amb l'ordre *docker compose ps -a* comprovarem que s'han creat els 4 dockers a partir del compose.



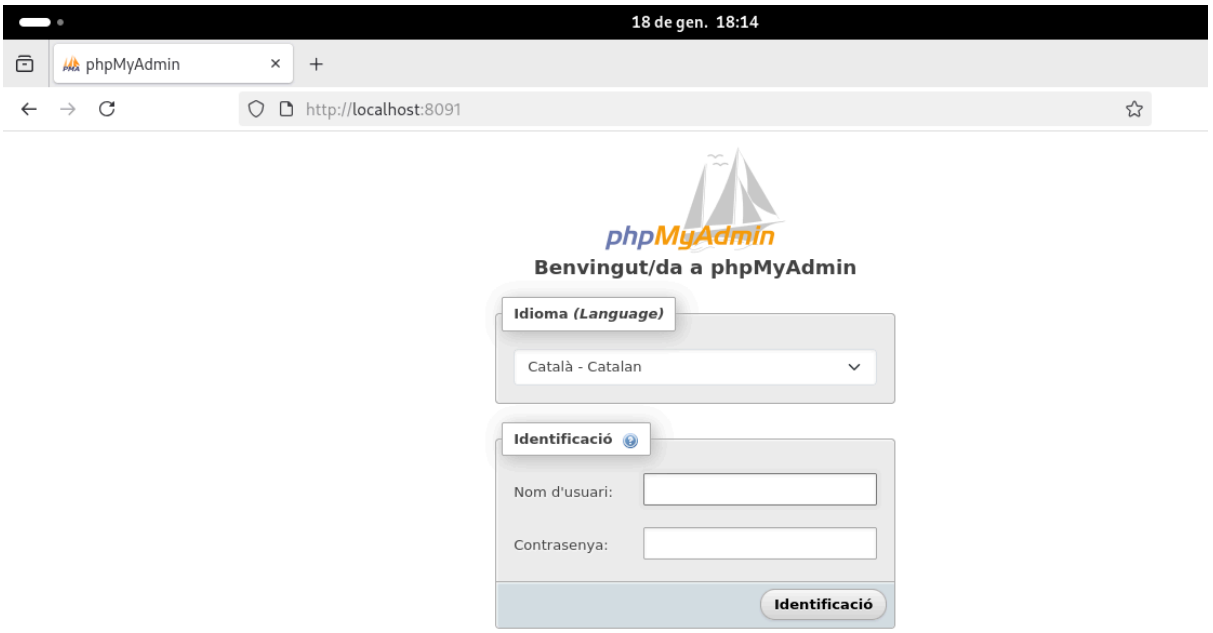
Amb l'ordre *sudo netstat -putan | grep 80* podem comprovar que els ports definits anteriorment estan escoltant en el nostre localhost, i tenen de nom de servei docker-proxy



Si accedim al localhost pel port 8090 (port exportat per nginx) veurem el contingut del fitxer index.html (informació de PHP)



I, si entrem al port 8091 (port exportat perPhpMyAdmin) accedirem al login de PhpMyAdmin



Pràctica 6.4

A partir del mateix Docker compose generat anteriorment, realitzarem una prova de connexió a una BD del contenidor creat anteriorment de mariadb.

Per connectar-nos a una BD hem de saber el nom o la IP del servidor amb MariaDB, en aquest cas, en ser un Docker, haurem de saber la IP del docker.

Per això, entrarem dins del docker amb l'ordre *docker exec -it <nomDocker> bash -l.* I un cop dins del docker executarem l'ordre *ip a* per recuperar la IP del Docker, en aquest cas 172.18.0.2

```
sebagu@sebagu-asix:/var/www/html/php/exercicis/practica6/6.3/lemp$ docker exec -it lemp-mariadb-1 bash -l
root@53164b201e05:/# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0@if29: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether d2:50:02:de:d8:74 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 172.18.0.2/16 brd 172.18.255.255 scope global eth0
       valid_lft forever preferred_lft forever
root@53164b201e05:/#
```

Recordem que tots els Dockers d'un mateix compose comparteixen subxarxa, per tant, tant Nginx com PhpMyAdmin tindran també una IP 172.18.0.X. Per entendre la distribució de xarxes i IPs de docker es pot consultar l'apartat "Configuració d'IPs de Dockers"

Aprofitant que estem dins del contenidor amb MariaDB, crearem la BD aula310

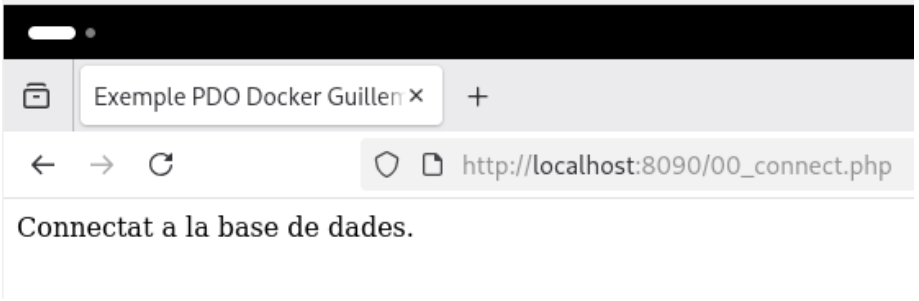
```
root@53164b201e05:/# mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 3
Server version: 10.9.8-MariaDB-1:10.9.8+maria~ubu2204 mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> create database aula310;
Query OK, 1 row affected (0.003 sec)

MariaDB [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| aula310            |
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.002 sec)

MariaDB [(none)]>
```

Seguidament, modificarem el fitxer 00_connect.php creat a l'estructura de directoris de la pràctica 6.3 i escriurem un codi per provar la connexió a la BD.

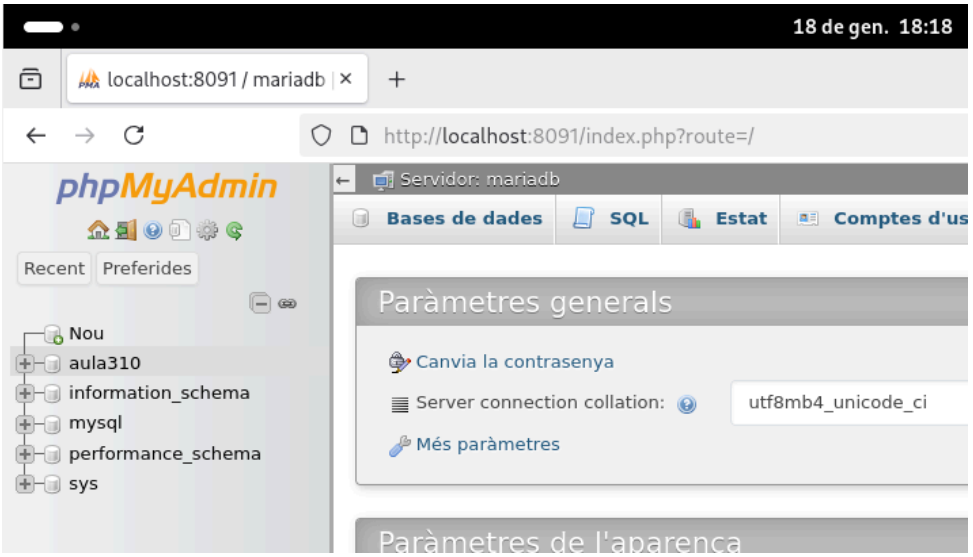
A servername, enlloc d'especificar "localhost" com hem fet fins ara, especificarem la IP del docker abans trobada.

```
sebagu@sebagu-asix: /var/ww
 GNU nano 8.4                                  00_connect.php
<!DOCTYPE html>
<html lang="ca">
    <head>
        <title>Exemple PDO Docker Guillem Serrat</title>
        <meta http-equiv="Content-type" content="text/html; charset=UTF-8">
    </head>
    <body>
        <?php
            // Definim els paràmetres per realitzar la connexió
            $servername = "172.18.0.2";      // El servidor on ens connectarem
            $username = "root";              // L'usuari de MariaDB
            $password = "fjeclot";           // La contrasenya de l'usuari
            $dbname = "aula310";             // La base de dades que farem servir

            try {
            // Definim un nou objecte de la classe PDO amb els atributs: host al que ens connectarem, la base de dades, el nom d'usuari i contrasenya.
            $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);

            // A través de la funció setAttribute, agafem el tipus d'error en cas de que n'hi hagi algun
            $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

            echo "Connectat a la base de dades.";

            } catch(PDOException $e) {                          // En cas de que hi hagi un error, PHP llença una PDOException, i la variable $
                echo "No es pot connectar. Motiu: " . $e->getMessage();     // Es mostra el missatge
            }

            $conn = null;                                       // Tanca la connexió amb la BBDD
        ?>
    </body>
</html>
```

I per últim, si entrem a localhost:8090 (port del Nginx) i obrim el fitxer 00_connect.php, comprovarem que ens hem connectat a la BD de forma exitosa



A més, dins de PhpMyAdmin (port 8091), comprovem que la BD aula310 existeix



Pràctica 6.5

Un cop hem vist com funciona Docker Compose, farem servir la mateixa eina per allotjar el projecte de PHP + MariaDB en Dockers.

Aprofitarem l'estructura de directoris de la pràctica anterior, replicant-los a un nou directori i copiarem tot el contingut del projecte de PHP + MariaDB dins el directori php-files (fitxers que presenta nginx)

```
sebagu@sebagu-asix: /var/wv
sebagu@sebagu-asix:/var/www/html/php/exercicis/practica6/6.5/lemp/php-files$ cp -r /var/www/html/php/exercicis/pj_php_mariadb/* .
sebagu@sebagu-asix:/var/www/html/php/exercicis/practica6/6.5/lemp/php-files$ ls
administracioDB  codis_pj_php_mariadb.zip  condicions  formularis  index.html  pdf  php
sebagu@sebagu-asix:/var/www/html/php/exercicis/practica6/6.5/lemp/php-files$
```

A continuació, ja que hem copiat tota l'estructura, haurem de modificar alguns aspectes del nou fitxer docker-compose:

- El nom dels contenidors
  - No poden existir dos dockers amb el mateix nom, per tant afegirem "_pj" al nom de tots els contenidors
- Els ports exposats
  - Els ports de l'actual compose ja s'estan fent servir pels dockers de la pràctica 6.3 i per tant els hem de modificar
    - Nginx exportarà el port 8095
    - PhpMyAdmin exportarà el port 8096

exemple.php | docker-compose.yml

C: > Users > Usuario > Documents > PHP > exercicis > practica6 > 6.5 > lemp > docker-compose.yml

```yaml
# version: '3.8' # L'atribut version ja no és necessari en l'última versió de docker

services:

  php_pj:
    build:
      dockerfile: php-dockerfile #Construim el Docker a partir de la imatge pròpia
    volumes:
      - './php-files:/var/www/html' # Definim un volum. Els documents del directori ./php-files del host es compartiran amb el directori /var/www/html del Docker
    depends_on:
      - mariadb_pj # Requerim que el servei mariadb estigui operatiu

  nginx_pj:
    image: nginx:latest # Utilitzem la imatge oficial de nginx, la última versió
    ports:
      - 8095:80 # El port 8095 del host dirigirà al port 80 del Docker
    links: # Avui en dia no cal especificar-ho, ja que docker compose ja crea una xarxa interna
      - 'php_pj' # Permet que el Docker Nginx es pugui comunicar amb el Docker "php"
    volumes:
      - './php-files:/var/www/html' # El contingut de ./php-files del host es compartirà amb el directori /var/www/html del Docker
      - './nginx-conf:/etc/nginx/conf.d' # El contingut de ./nginx-conf del host es compartirà amb el directori /etc/nginx/conf.d del Docker
    depends_on:
      - php_pj # Requerim que el servei php estigui operatiu

  mariadb_pj: # No exportem cap port ja que la BD no cal que sigui accesible des de l'exterior, sinó que únicament des dels altres Dockers
    image: mariadb:10.9 # Fem servir la imatge oficial de mariadb
    environment:
      MYSQL_ROOT_PASSWORD: fjeclot # Definim la contrasenya de root de MariaDB
    volumes:
      - mysqldata:/var/lib/mysql # El contingut de /var/lib/mysql es desarà a un volum intern de Docker anomenat mysqldata (NO ÉS UN DIRECTORI DEL PROJECTE, ÉS UN VOLUM INTERN DE DOCKER)

  phpmyadmin_pj:
    image: phpmyadmin/phpmyadmin:latest # Fem servir la imatge de PHPMyAdmin
    ports:
      - 8096:80 # El port 8096 del host dirigirà al port 80 del Docker
    environment:
      PMA_HOST: mariadb_pj # Indiquem el hostname de la BD, en aquest cas el nom del docker
    depends_on:
      - mariadb_pj # Requerim que el servei mariadb estigui operatiu

# Definim els volums INTERNS de Docker
volumes:
  mysqldata: # Els arxius dels volumns interns de Docker s'acustuma a desar-se a /var/lib/docker/volumes
```

A continuació, dins de la configuració d'Nginx definim el nom del contenidor que pot treballar amb PHP. Per tant, hem de modificar el nom del contenidor a php_pj

```
server {
    listen 80 default_server; # Escoltem pel port 80, tot i que després el port 80 del Docker es converteix en el 8090 del host
    listen [::]:80 default_server;

    server_name localhost;

    root /var/www/html;
    index index.php index.html;

    location / {
        try_files $uri $uri/ /index.php?$args;
    }

    location ~* \.php$ {
        fastcgi_pass php_pj:9000;
        include fastcgi_params;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        fastcgi_param SCRIPT_NAME $fastcgi_script_name;
    }
}
```

Per últim, revisarem la IP del Docker amb MariaDB i canviarem la IP de les connexions a la BD del propi projecte

```php
<?php
    $servidor = "172.19.0.2";
    $usuari = "iot";
    $contrasenya = "iot";
    $nomDB = "Usuaris";
    $nomTaula = "Usuaris";
```

```php
<?php
    $servidor = "172.19.0.2";
    $usuari = "convidat";
    $contrasenya = "benvingut";
    $nomDB = "Usuaris";
    $nomTaula = "Usuaris";
```

Dins del projecte, depenent de les necessitats de l'aplicació fem consultes amb un usuari o un altre, però actualment aquests usuaris encara no hi són presents a MariaDB i cal crear-los

```
sebagu@sebagu-asix:/var/www/html/php/exercicis/practica6/6.5/lempPj$ docker exec -it lemppj-mariadb_pj-1 bash -l
root@62afd5884ac6:/# mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 4
Server version: 10.9.8-MariaDB-1:10.9.8+maria~ubu2204 mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

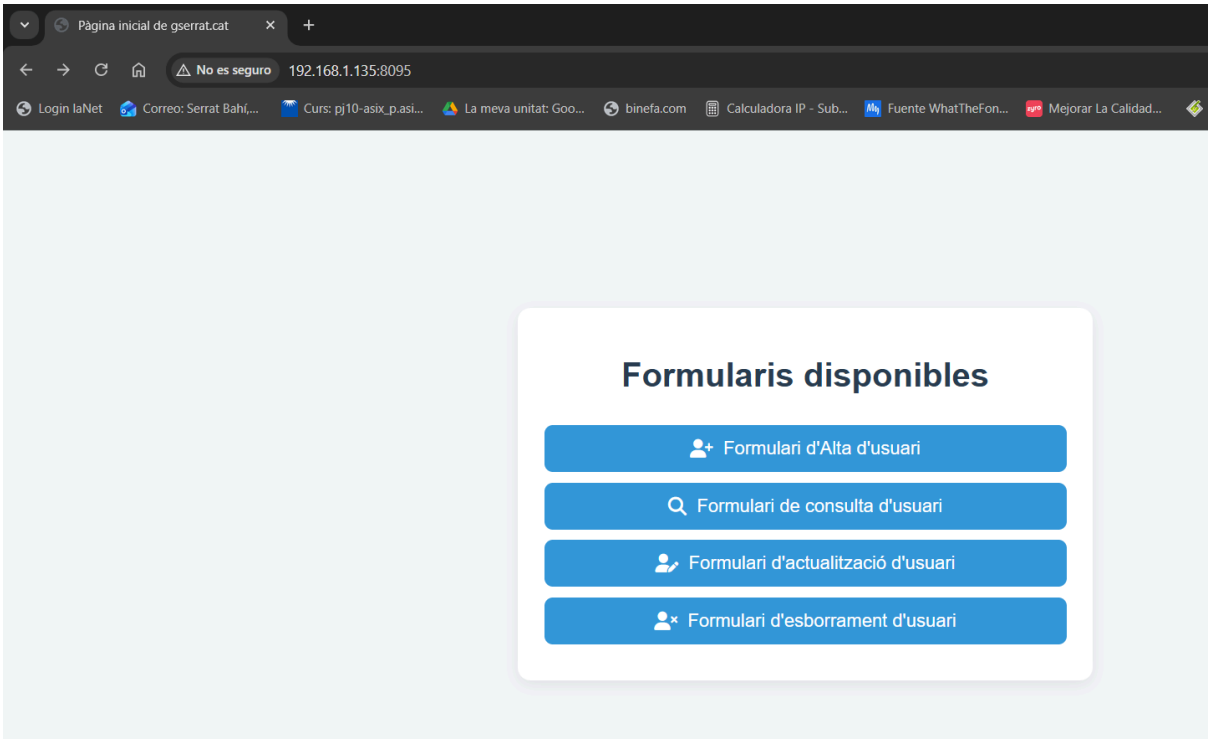
Fins ara, únicament permetíem accés als usuaris des del localhost, però com hara la BD i l'aplicació estan en dockers (hosts) diferents, permetrem l'accés des de qualsevol equip amb el símbol "%".

```
MariaDB [(none)]> CREATE USER 'iot'@'%' IDENTIFIED BY 'iot';
Query OK, 0 rows affected (0.004 sec)

MariaDB [(none)]> CREATE USER 'convidat'@'%' IDENTIFIED BY 'benvingut';
Query OK, 0 rows affected (0.005 sec)
```

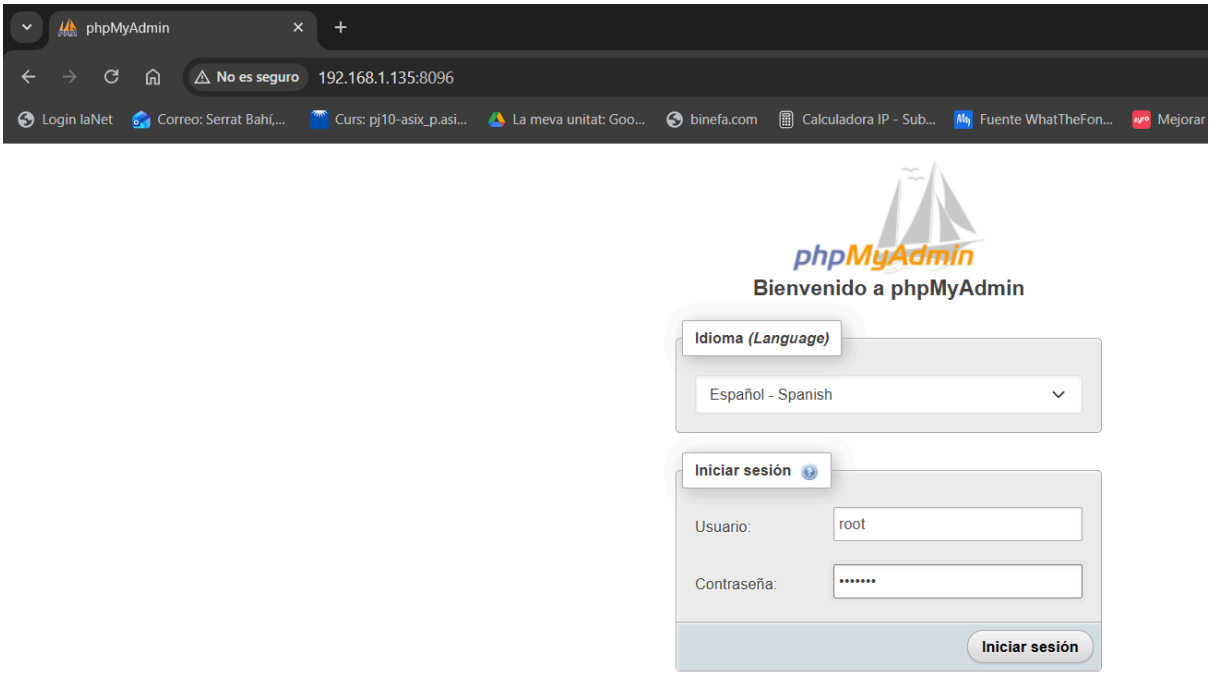
Per últim, un cop afegits els usuaris, en la creació de la BD quan els hi assignem permissos haurem de canviar "localhost" pel símbol "%" per permetre l'accés des de qualsevol equip, d'igual forma que en la creació dels mateixos

```
$permisosRW = "GRANT ALL PRIVILEGES ON Usuaris.* TO 'iot'@'%';";
$permisosR = "GRANT SELECT ON Usuaris.* TO 'convidat'@'%';";
```

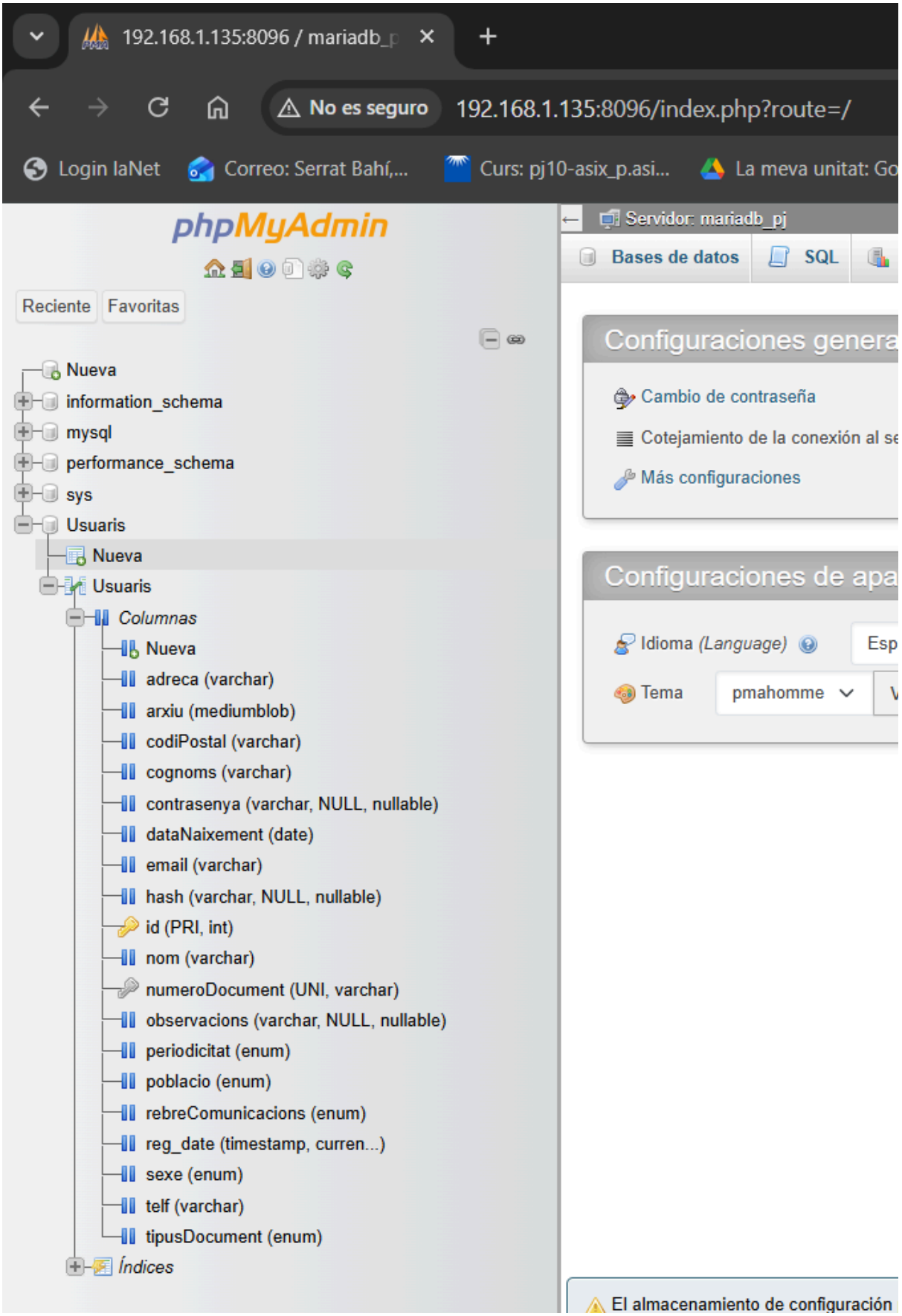
Un cop realitzades les tasques de posada en producció, si accedim a la màquina pel port 8095 accedirem a la pàgina inicial del projecte



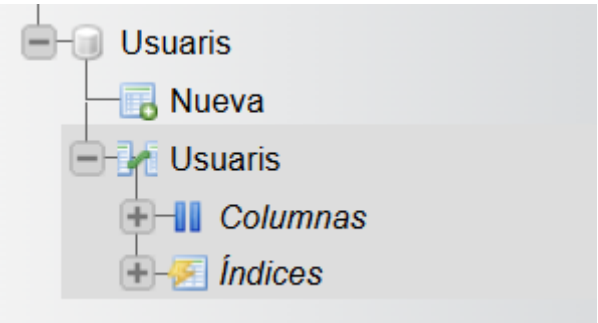
I si accedim al port 8096, al PhpMyAdmin, el qual fa servir les mateixes credencials de MariaDB



Dins de PhpMyAdmin, a la part lateral esquerra, trobem un desplegable amb les BD de MariaDB, on trobem la BD Usuaris, creada pel projecte



Si entrem dins de la taula Usuaris



Al menú lateral dret veurem tots els registres. En aquest cas en tenim un.

	id	tipusDocument	numeroDocument	nom	cognoms	sexe	dataNaixement	email	telf	poblacio	codiPostal	adreca
Editar Copiar Borrar	1	NIF	12345678Z	Guillem	Serrat	home	2000-01-01	guillem@gserrat.cat	630148965	Barcelona	08001	Cc

Migració i configuració del VPS

Migració dels Dockers

Per migrar els Dockers al VPS d'Azure únicament copiarem els directoris de la pràctica 6 al VPS. Gràcies a treballar amb Dockers, no cal instal·lar cap programari (a part dels necessaris per treballar amb ells), ja que els propis Dockers descarreguen les imatges i repliquen les configuracions.

Configuració dels Dockers

Configuració d'IPs dels Dockers

Un cop migrats els Dockers hem de revisar un aspecte molt important, les IPs d'aquests.

La subxarxa que té docker és la 172.17.0.0/16. Cada contenidor individual se li assigna una IP en aquella xarxa, començant per la .2. Per exemple:

- Docker 1 = 172.17.0.2
- Docker 2 = 172.17.0.3

En canvi, cada Docker Compose genera una xarxa privada entre tots els contenidors, començant pel 18, ja que el 17 és la xarxa de Dockers "individuals". Cada contenidor s'identifica amb el quart octet.

- Compose 1 = 172.18.0.0
    - Docker 1 = 172.17.0.2
    - Docker 2 = 172.17.0.3
    - Docker 3 = 172.17.0.4
- Compose 2 = 172.19.0.0
    - Docker 1 = 172.18.0.2
    - Docker 2 = 172.18.0.3
    - Docker 3 = 172.18.0.4

Totes les IP de xarxes i hosts s'assignen de manera incremental, és a dir, el que primer s'executa és el que rep la primera IP.

Això és important ja que actualment al meu VPS tinc un MediaWiki, la qual fa servir la xarxa 172.18.0.0/16. Això vol dir que:

- El Docker de la pràctica 6.2 farà servir la xarxa
    - 172.17.0.0/16 (ja que és un docker individual)
- MediaWiki farà servir la xarxa
    - 172.18.0.0/16
- El Docker de les pràctiques 6.3 i 6.4 faran servir la xarxa
    - 172.19.0.0/16
- El Docker de la pràctica 6.5 farà servir la xarxa
    - 172.20.0.0/16

Originalment, la distribució d'IPs era la següent:

- La xarxa 172.18.0.0/16 (actualment de MediaWiki) era de la pràctica 6.3 i 6.4
- La xarxa 172.19.0.0/16 (actualment de les pràctiques 6.3 i 6.4) era de la pràctica 6.5

Tot i això, com a administradors de sistemes no podem dependre d'aquesta suposició, per tant, enlloc de definir la IP del docker (que pot canviar), definirem directament el nom del propi Docker, el qual no canviarà mai

Els noms dels docker es crean segons la següent estructura
<nomDirectori>_<nomServei>_<número>

Per exemple:

- lemp-nginx-1
- lemp-mariadb-1
- lemp-php-1
- lemp-phpmyadmin-1

En el cas de la pràctica 6.3 i 6.4, modificarem el fitxer 00_connect.php i especificarem el nom del host amb el nom del contenidor

```php
<?php
    // Definim els paràmetres per realitzar la connexió
    $servername = "lemp-mariadb-1";      // El servidor on ens connectarem
    $username = "root";                  // L'usuari de MariaDB
    $password = "fjeclot";               // La contrasenya de l'usuari
    $dbname = "aula310";                 // La base de dades que farem servir
```

En el cas de la pràctica 6.5, modificarem tots aquells fitxers del projecte de MariaDB que es connectin a la BD, i canviarem el seu host pel docker anomenat "lemppj-mariadb_pj-1"

```
  GNU nano 8.4
<?php
    $servidor = "lemppj-mariadb_pj-1";
    $usuari = "iot";
    $contrasenya = "iot";
    $nomDB = "Usuaris";
    $nomTaula = "Usuaris";
```

Ports exposats dels Dockers

- Pràctica 6.2
    - Port 8089: Pàgina Inicial
- Pràctica 6.3 i 6.4
    - Port 8090: Pàgina Inicial
    - Port 8091: PhpMyAdmin
- Pràctica 6.5
    - Port 8095: Pàgina inicial del projecte
    - Port 8096: PhpMyAdmin

Gestió de les BD de les pràctiques 6.3, 6.4 i 6.5

Creació de la BD aula310 de les pràctiques 6.3 i 6.4

Degut a que al migrar els Dockers no hem migrat els volums interns de Docker, haurem de tornar a crear la BD per la pràctica 6.3 i 6.4

```
root@48fbc55b54ee:/# mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 4
Server version: 10.9.8-MariaDB-1:10.9.8+maria~ubu2204 mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> create database aula310;
Query OK, 1 row affected (0.000 sec)

MariaDB [(none)]>
```

Creació d'usuaris per la pràctica 6.5

Degut a que al migrar els Dockers no hem migrat els volums interns de Docker, haurem de tornar els usuaris de MariaDB per accedir a les BD

La BD del projecte es pot crear a partir dels codis PHP

```
MariaDB [(none)]> CREATE USER 'iot'@'%' IDENTIFIED BY 'iot';
Query OK, 0 rows affected (0.002 sec)

MariaDB [(none)]> CREATE USER 'convidat'@'%' IDENTIFIED BY 'benvingut';
Query OK, 0 rows affected (0.002 sec)

MariaDB [(none)]>
```

Configuració d'Nginx

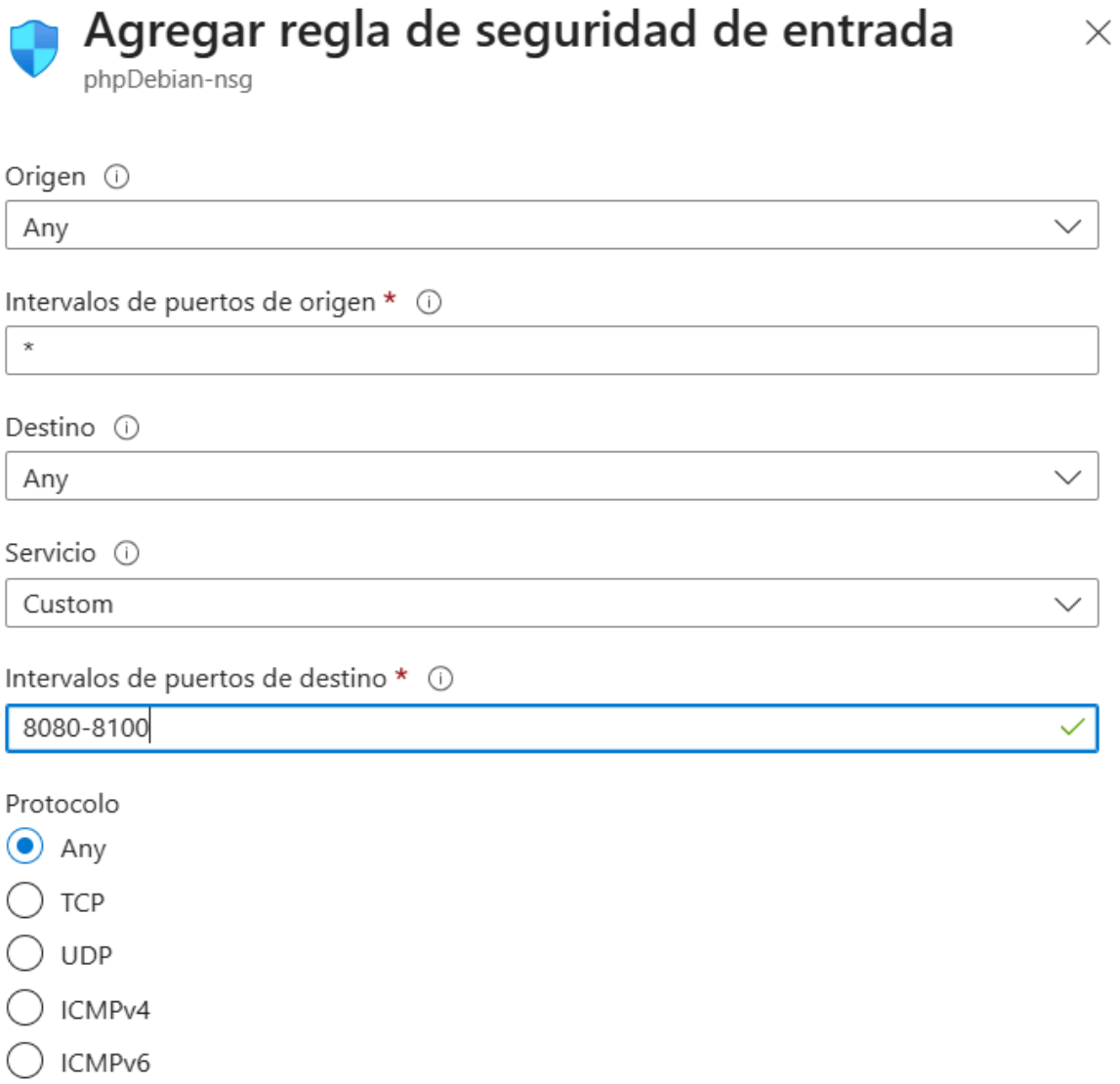
Nginx té un límit de mida de fitxer per pujar-lo al servidor, per això hem d'afegir la directiva *client_max_body_size* a la configuració d'Nginx i especificar una mida, per exemple 100 MB

```
guseba@phpDebianAzure: /v
  GNU nano 8.4                                    nginx.conf
server {
    listen 80 default_server; # Escoltem pel port 80, tot i que després el port 80 del Docker es converteix en el 8090 del host
    listen [::]:80 default_server;

    server_name localhost;

    root /var/www/html;
    index index.php index.html;

    # Limit màxim de pujada d'arxiu
    client_max_body_size 100M;

    location / {
        try_files $uri $uri/ /index.php?$args;
    }

    location ~* \.php$ {
        fastcgi_pass php_pj:9000;
        include fastcgi_params;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        fastcgi_param SCRIPT_NAME $fastcgi_script_name;
    }
}
```

Configuració del NSG d'Azure

Per més informació sobre què són i per què és important configurar els NSG d'Azure, es pot consultar aquest article de la Wiki de gserrat.cat

En aquest cas, ja que fem servir diversos ports (i en futures pràctiques farem servir més), desbloquejarem els rang de ports del 8080 al 8100, així disposem de la possibilitat d'exportat fins a 20 dockers



Configuració de noms de domini amb Nominalia

Per aquesta pràctica es definiran noms de domini que redirigiran directament al docker, sense necessitat d'especificar el nom genèric del VPS i el port del docker.

Per això crearé els següents nom de domini

- docker6-2.gserrat.cat
- docker6-3.gserrat.cat
- docker6-5.gserrat.cat
- admin.docker6-5.gserrat.cat

I dins de Nominalia, el meu proveïdor de domini, crearé els següents registres DNS

- Registres A
  - docker6-2.gserrat.cat -> IP del VPS
  - docker6-3.gserrat.cat -> IP del VPS
  - docker6-5.gserrat.cat -> IP del VPS
  - admin.docker6-5.gserrat.cat -> IP del VPS
- Registres CNAME
  - www.docker6-2.gserrat.cat -> docker6-2.gserrat.cat
  - www.docker6-3.gserrat.cat -> docker6-3.gserrat.cat
  - www.docker6-5.gserrat.cat -> docker6-5.gserrat.cat
  - www.admin.docker6-5.gserrat.cat -> admin.docker6-5.gserrat.cat

Configuració dels hosts virtuals d'Apache

Instal·lació de mòduls d'apache necessaris

L'objectiu dels hosts virtuals és accedir directament al docker, sense necessitat d'especificar el nom genèric del VPS i el port del docker.

Per realitzar aquesta feina amb Apache requerim de la habilitació dels següents mòduls:

- proxy
- proxy_http

Tots dos es poden habilitar amb l'ordre *sudo a2enmod <nomModul>*

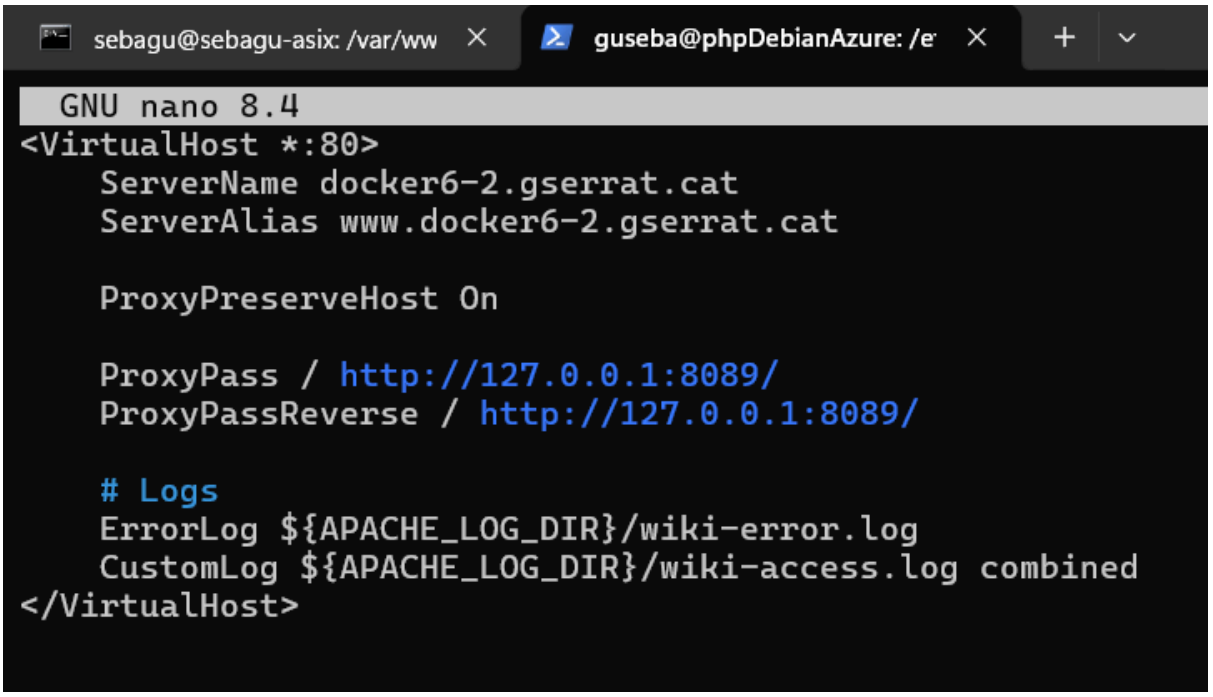
Host virtual de la pràctica 6.2

Per la pràctica 6.2 definirem el nom de domini docker6-2.gserrat.cat i l'àlies www.docker6-2.gserrat.cat.

En cas d'accedir al servidor amb algun d'aquests dos noms de domini, redirigirem a l'usuari a "http://127.0.0.1:8089", és a dir, al localhost al port 8089, on s'està exportant el docker de la pràctica 6.2.

La "/" indica que accedim a l'arrel del directori, és a dir, docker6-2.gserrat.cat i no docker6-2.gserrat.cat/altre.html

Això ho realitzarem especificant la directiva ProxyPass, i mitjançant la directiva ProxyPassReverse reescriu la capçalera de resposta i redireccionament per tal de que mostri el nom de domini



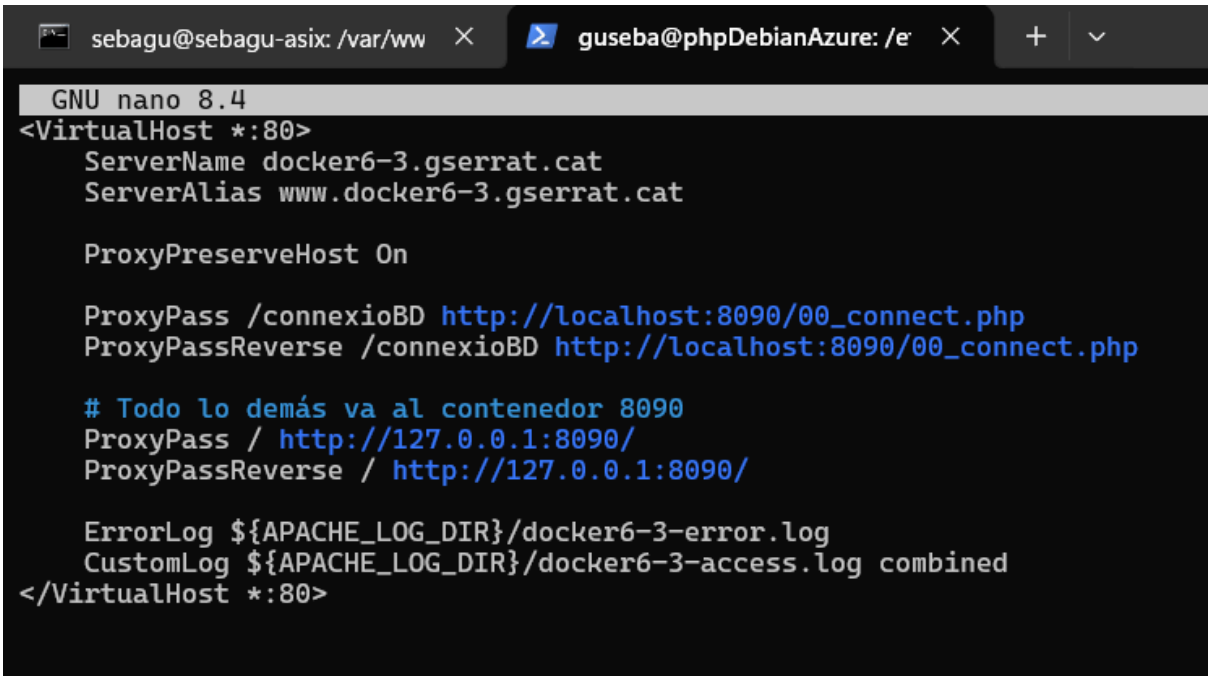
Host virtual de la pràctica 6.3 i 6.4

Per la pràctica 6.3 i 6.4 definirem el nom de domini docker6-3.gserrat.cat i l'àlies www.docker6-3.gserrat.cat.

En cas d'accedir al servidor amb algun d'aquests dos noms de domini, redirigirem a l'usuari a "http://127.0.0.1:8090", és a dir, al localhost al port 8090, on s'està exportant el docker de la pràctica 6.3 i 6.4.

En canvi, si especifiquem la ruta docker6-3.gserrat.cat/connexioBD, redirigirem també al propi docker però directament al fitxer 00_connect.php

Això ho realitzarem especificant la directiva ProxyPass, i mitjançant la directiva ProxyPassReverse reescriu la capçalera de resposta i redireccionament per tal de que mostri el nom de domini

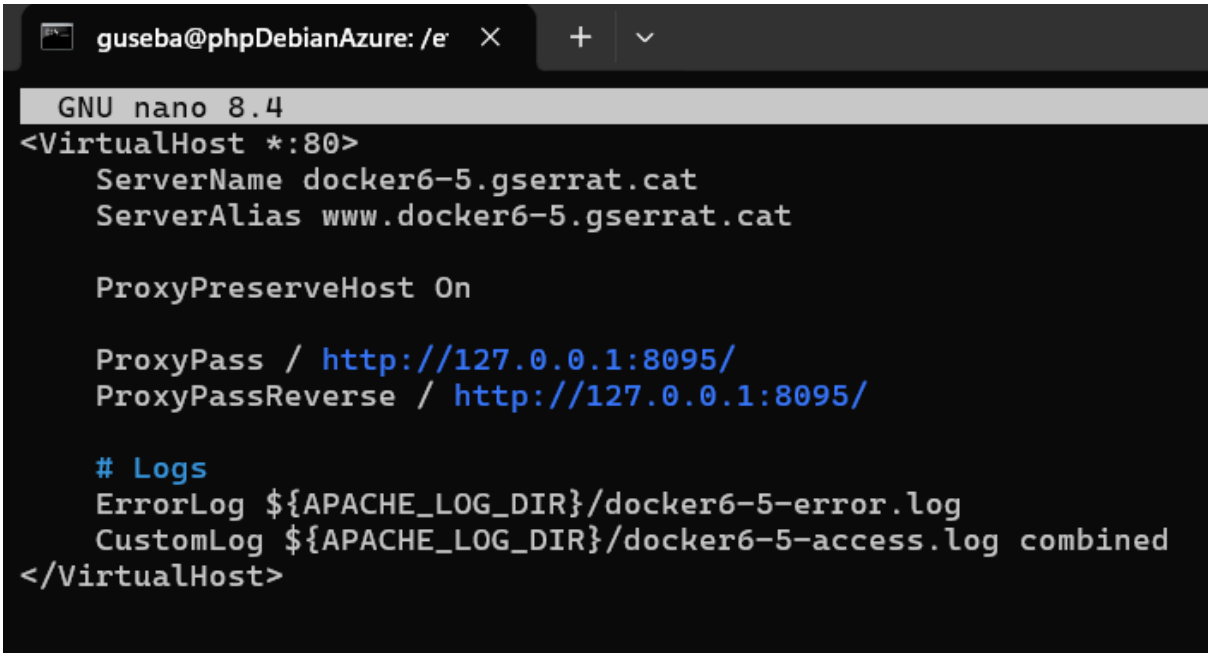


Host virtual de la pràctica 6.5 - Pàgina inicial del projecte

Per la pàgina inicial del projecte de la pràctica 5 definirem el nom de domini docker6-5.gserrat.cat i l'àlies www.docker6-5.gserrat.cat.

En cas d'accedir al servidor amb algun d'aquests dos noms de domini, redirigirem a l'usuari a "http://127.0.0.1:8096", és a dir, al localhost al port 8096, on s'està exportant el docker de PhpMyAdmin

Això ho realitzarem especificant la directiva ProxyPass, i mitjançant la directiva ProxyPassReverse reescriu la capçalera de resposta i redireccionament per tal de que mostri el nom de domini



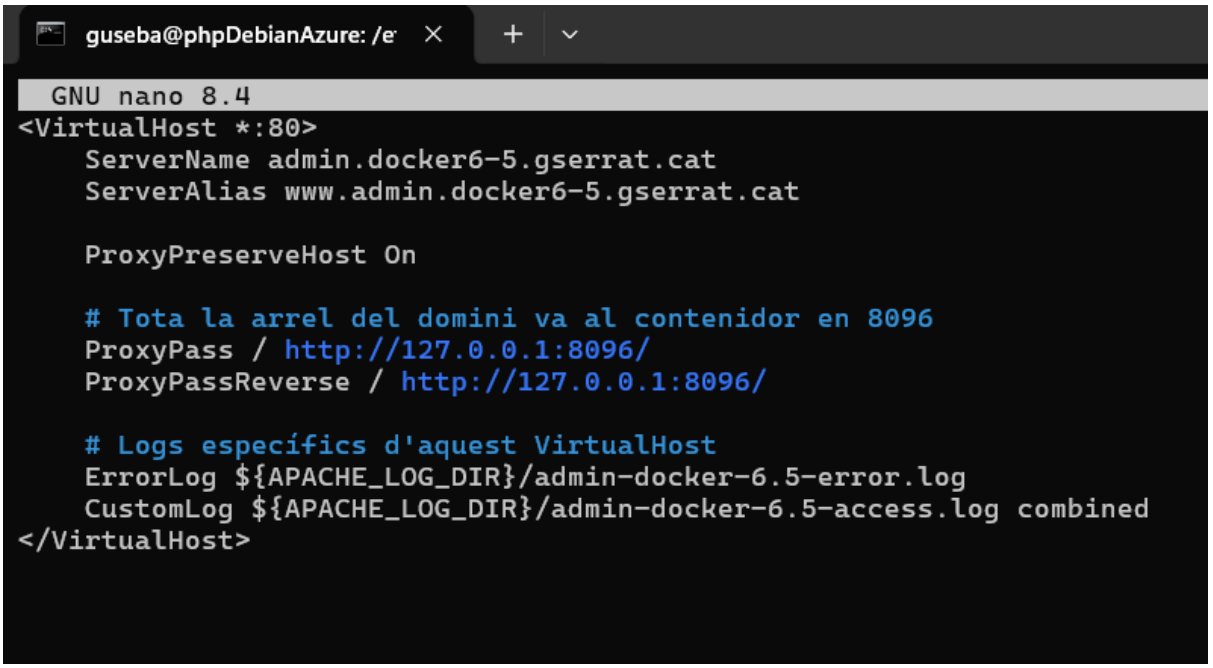
Host virtual de la pràctica 6.5 - PhpMyAdmin

Pel PhpMyAdmin de la pràctica 5 definirem el nom de domini admin.docker6-5.gserrat.cat i l'àlies www.admin.docker6-5.gserrat.cat.

En cas d'accedir al servidor amb algun d'aquests dos noms de domini, redirigirem a l'usuari a "http://127.0.0.1:8096", és a dir, al localhost al port 8096, on s'està exportant el docker de PhpMyAdmin

Això ho realitzarem especificant la directiva ProxyPass, i mitjançant la directiva ProxyPassReverse reescriu la capçalera de resposta i redireccionament per tal de que mostri el nom de domini

Creem un nou nom de domini i host virtual per PhpMyAdmin ja que al generar-se rutes absolutes en aquest servei (a l'igual que MediaWiki), necessitem un nom de domini que actuï com a arrel, sense subrutes com docker6-5.gserrat.cat/**phpmyadmin**



Certificació SSL

Preparació per la certificació SSL

Un cop tenim tots els hosts virtuals haurem de certificar-los per poder fer servir HTTPS.

Per això hem de:

- Instal·lar certbot amb l'ordre *sudo apt install certbot python3-certbot-apache -y,* l'eina que permet emetre certificats gratuits
- Verificar que el domini ja està creat al DNS
- Assegurar que el host virtual està activat (*sudo a2ensite <hostVirtual>*

Domini docker6-2.gserrat.cat

Començarem certificant el domini docker6-2.gserrat.cat (i també el seu àlies www.docker6-2.gserrat.cat) amb l'ordre *sudo certbot --apache -d docker6-2.gserrat.cat -d www.docker6-2.gserrat.cat*

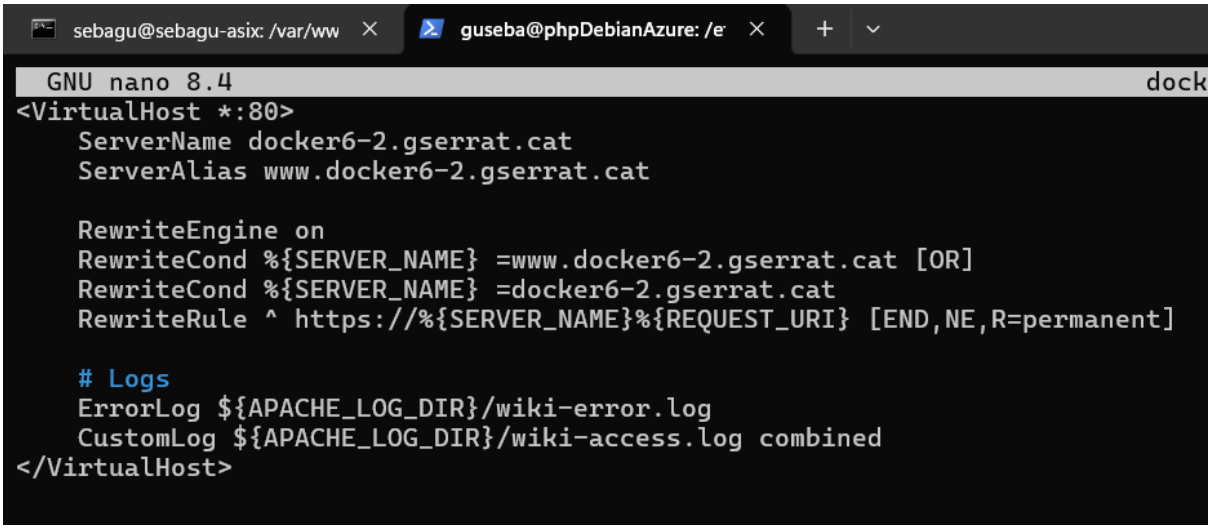
Un cop certificat ens generarà un nou host virtual que escoltarà pel port 443 d'Apache i on tindrà els mòduls de certificació en HTTPS

```
guseba@phpDebianAzure:/etc/apache2/sites-available$ cat docker6-2-le-ssl.conf
<IfModule mod_ssl.c>
<VirtualHost *:443>
    ServerName docker6-2.gserrat.cat
    ServerAlias www.docker6-2.gserrat.cat

    ProxyPreserveHost On

    ProxyPass / http://127.0.0.1:8089/
    ProxyPassReverse / http://127.0.0.1:8089/

    # Logs
    ErrorLog ${APACHE_LOG_DIR}/wiki-error.log
    CustomLog ${APACHE_LOG_DIR}/wiki-access.log combined

Include /etc/letsencrypt/options-ssl-apache.conf
SSLCertificateFile /etc/letsencrypt/live/docker6-2.gserrat.cat/fullchain.pem
SSLCertificateKeyFile /etc/letsencrypt/live/docker6-2.gserrat.cat/privkey.pem
</VirtualHost>
</IfModule>
guseba@phpDebianAzure:/etc/apache2/sites-available$
```

Al host virtual sense certificat ens generarà unes directives de redirecció a HTTPS, així si un usuari entra per HTTP, el host virtual redireccionarà a HTTPS

També és possible deshabilitar el lloc virtual per evitar connexions HTTP, però no és recomanable ja que a l'usuari li apareixerà com que el lloc web no és accessible. La millor pràctica és mantenir el lloc web HTTP actiu amb una redirecció a HTTPS

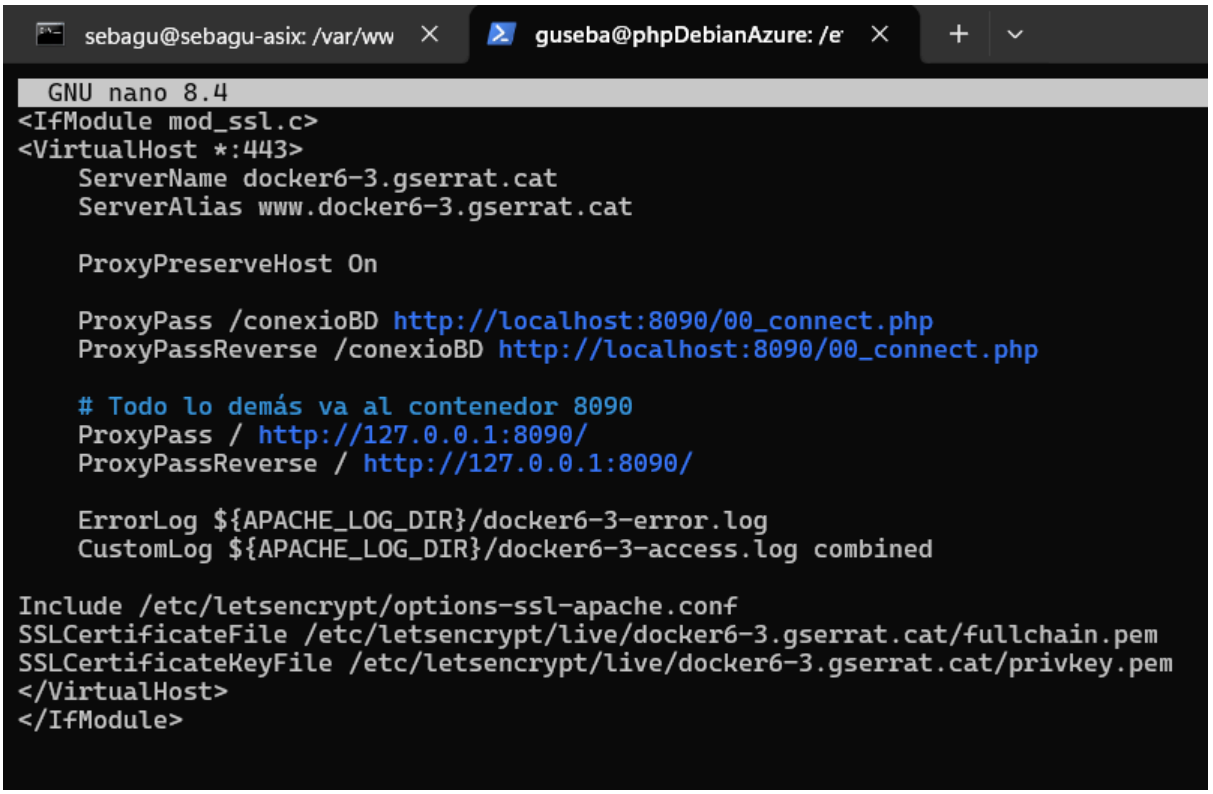
Tot i això, el codi original es mantindrà dins del host virtual. És recomanable retirar tot el codi i deixar únicament la redirecció.



Domini docker6-3.gserrat.cat

A continuació certificarem el domini docker6-3.gserrat.cat (i també el seu àlies www.docker6-3.gserrat.cat) amb l'ordre *sudo certbot --apache -d docker6-3.gserrat.cat -d www.docker6-3.gserrat.cat*

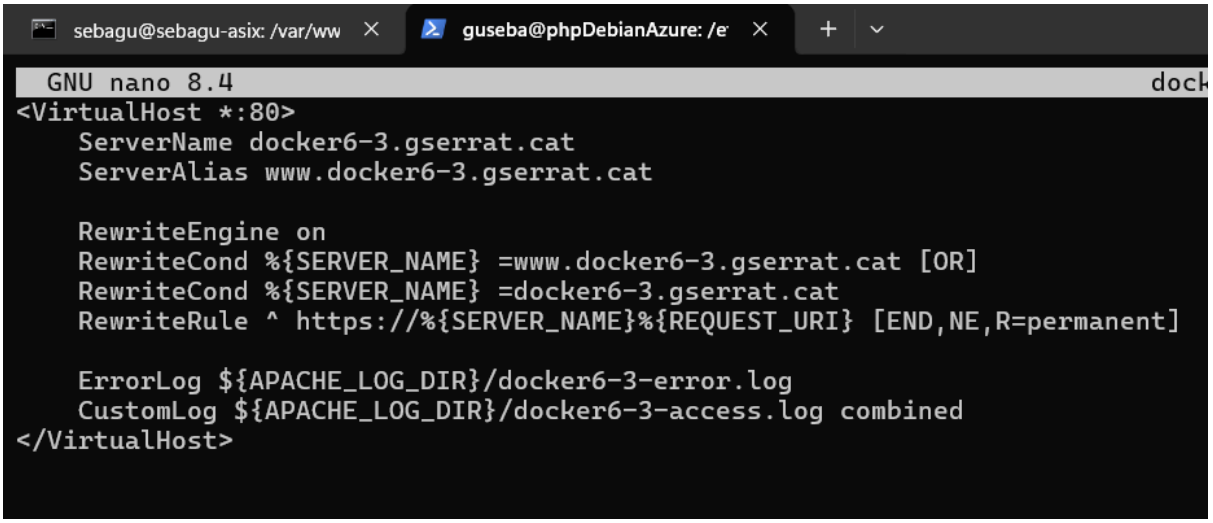
Un cop certificat ens generarà un nou host virtual que escoltarà pel port 443 d'Apache i on tindrà els mòduls de certificació en HTTPS



Al host virtual sense certificat ens generarà unes directives de redirecció a HTTPS, així si un usuari entra per HTTP, el host virtual redireccionarà a HTTPS

També és possible deshabilitar el lloc virtual per evitar connexions HTTP, però no és recomanable ja que a l'usuari li apareixerà com que el lloc web no és accessible. La millor pràctica és mantenir el lloc web HTTP actiu amb una redirecció a HTTPS

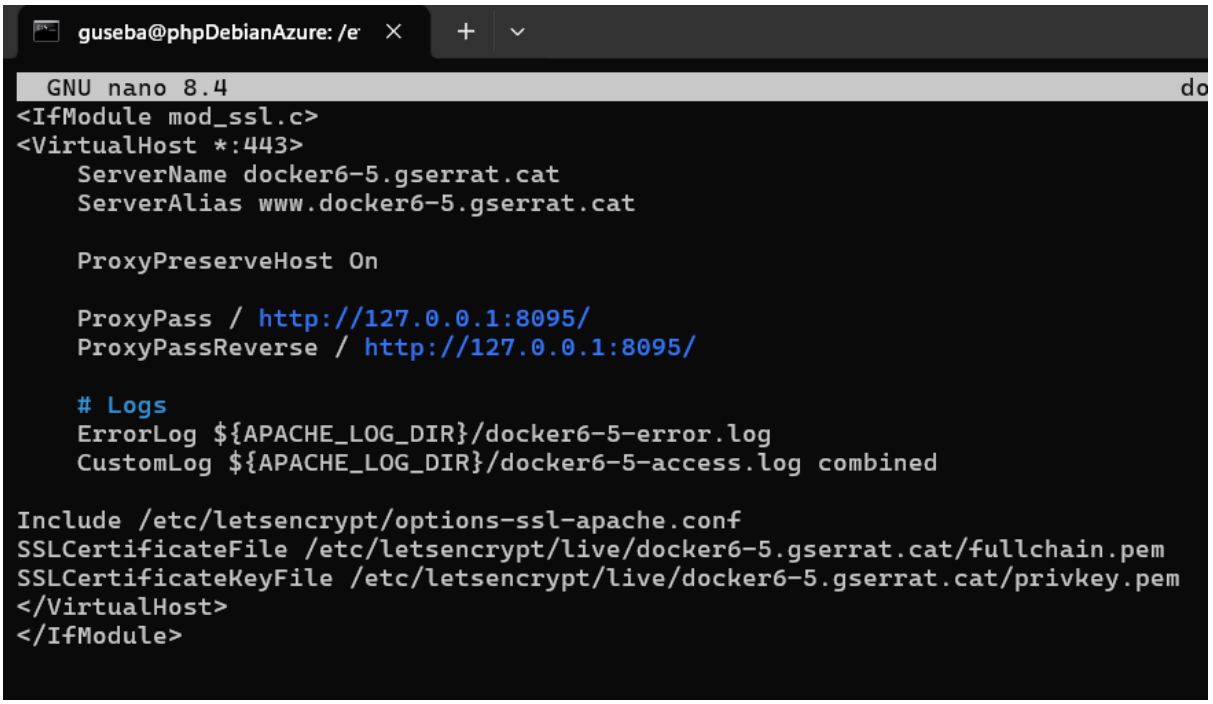
Tot i això, el codi original es mantindrà dins del host virtual. És recomanable retirar tot el codi i deixar únicament la redirecció.



Domini docker6-5.gserrat.cat

A continuació certificarem el domini docker6-5.gserrat.cat (i també el seu àlies www.docker6-5.gserrat.cat) amb l'ordre *sudo certbot --apache -d docker6-5.gserrat.cat -d www.docker6-5.gserrat.cat*

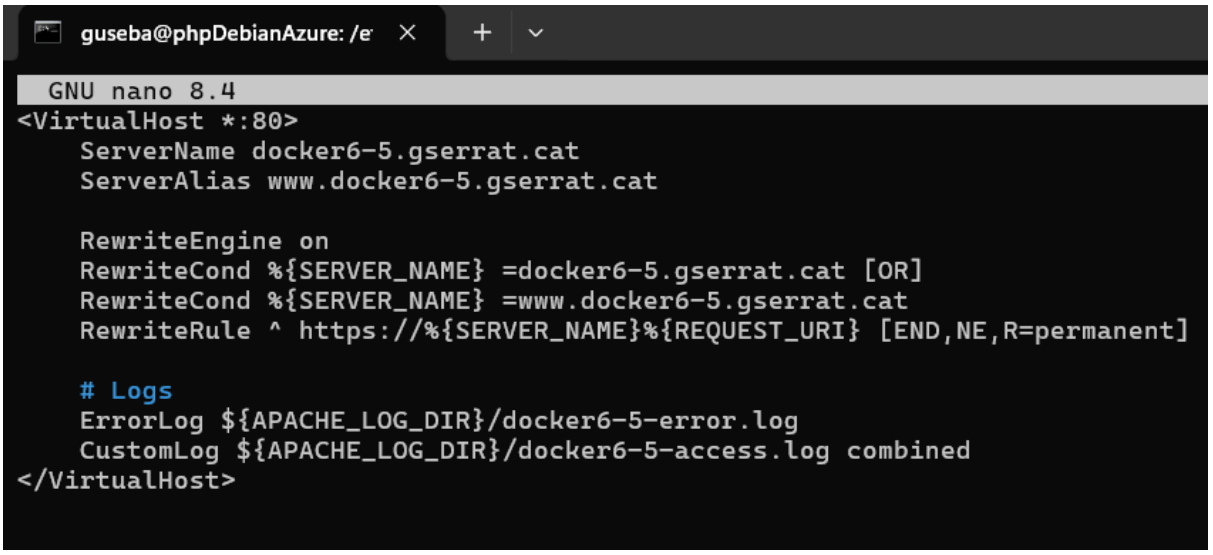
Un cop certificat ens generarà un nou host virtual que escoltarà pel port 443 d'Apache i on tindrà els mòduls de certificació en HTTPS



Al host virtual sense certificat ens generarà unes directives de redirecció a HTTPS, així si un usuari entra per HTTP, el host virtual redireccionarà a HTTPS

També és possible deshabilitar el lloc virtual per evitar connexions HTTP, però no és recomanable ja que a l'usuari li apareixerà com que el lloc web no és accessible. La millor pràctica és mantenir el lloc web HTTP actiu amb una redirecció a HTTPS

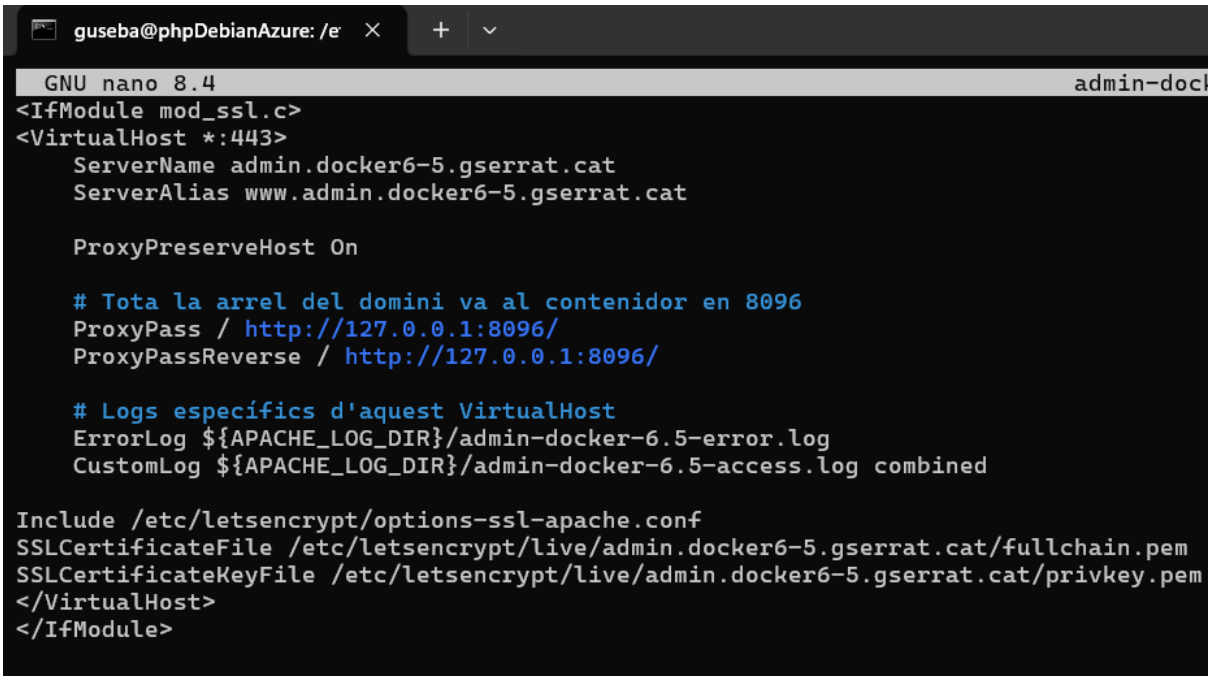
Tot i això, el codi original es mantindrà dins del host virtual. És recomanable retirar tot el codi i deixar únicament la redirecció.



Domini admin.docker6-5.gserrat.cat

Per últim certificarem el domini admin.docker6-5.gserrat.cat (i també el seu àlies www.admin.docker6-5.gserrat.cat) amb l'ordre *sudo certbot --apache -d admin.docker6-5.gserrat.cat -d www.admin.docker6-5.gserrat.cat*

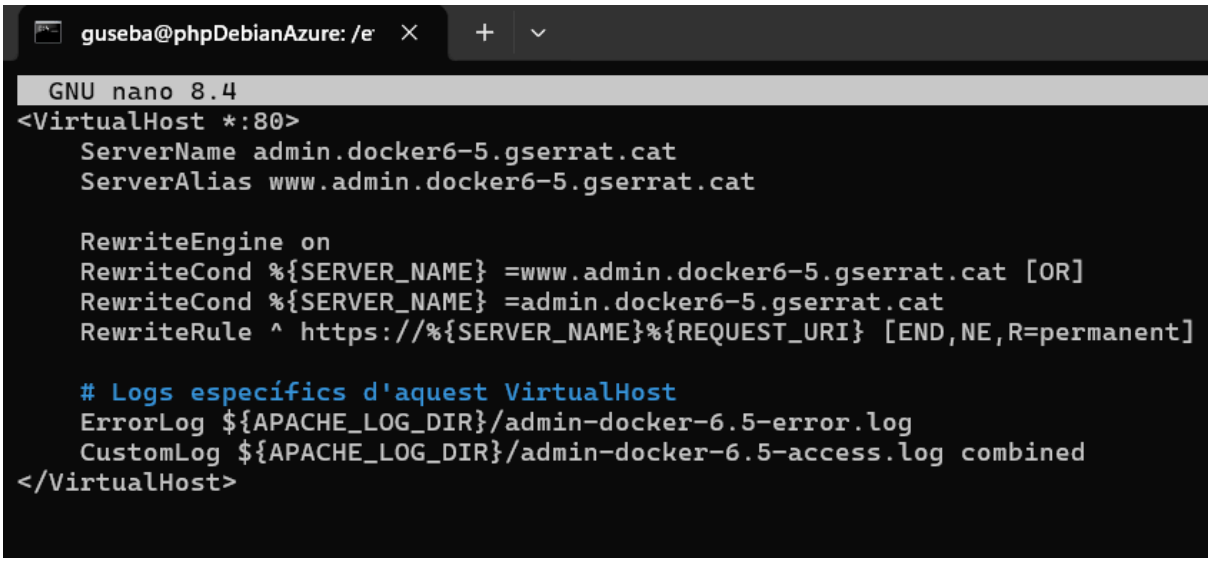
Un cop certificat ens generarà un nou host virtual que escoltarà pel port 443 d'Apache i on tindrà els mòduls de certificació en HTTPS



Al host virtual sense certificat ens generarà unes directives de redirecció a HTTPS, així si un usuari entra per HTTP, el host virtual redireccionarà a HTTPS

També és possible deshabilitar el lloc virtual per evitar connexions HTTP, però no és recomanable ja que a l'usuari li apareixerà com que el lloc web no és accessible. La millor pràctica és mantenir el lloc web HTTP actiu amb una redirecció a HTTPS

Tot i això, el codi original es mantindrà dins del host virtual. És recomanable retirar tot el codi i deixar únicament la redirecció.



Persistència de certificats

Els certificats SSL expedits per Let's Encrypt tenen una caducitat de 90 dies. Tot i això podem automatitzar la renovació d'aquests amb les següents ordres:

- *sudo systemctl enable certbot.timer*
- *sudo systemctl start certbot.timer*

D'aquesta manera, un cop els certificats caduquin, el servei de certbot expedirà un nou certificat