Pràctica 7:
Wordpress emprant Dockers

Guillem Serrat Bahí

Index

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 Wordpress

Documentació en format Wiki
https://wiki.gserrat.cat/index.php/Pr%C3%A0ctica_7_Wordpress_emprant_dockers

Introducció

En aquesta pràctica desplegarem el CMS Wordpress mitjançant Dockers.

Primerament, crearem dos contenidors manualment, un per Wordpress i l'altre per la BD. Enllaçarem els dos contenidors i exportarem un port en concret.

A continuació, realitzarem el mateix però amb Docker Compose, una funcionalitat de Docker que permet executar diversos contenidors i configuracions amb un sol fitxer de configuració.

Per últim, es posarà en producció un Wordpress al VPS d'Azure, per tal de ser accesible des d'internet

Pràctica 7.1

Creació del contenidor amb MariaDB

Per treballar amb Wordpress necessitem tenir una BD on desar tota la informació (articles, pàgines, enllaços, etc).

Per això, primerament crearem un contenidor de MariaDB amb les següents característiques:

- Nom
  - El nom del contenidor serà wordpress-db
- Volums
  - Crearem un únic volum el qual serà un volum intern de Docker anomenat “wordpress-db” que replicarà la informació al directori del Docker /var/lib/mysql (directori on es desen les BD)
  - En aquest cas no especifiquem tipus i per tant el tipus per defecte és “volume”
  - Cal recordar que els volums interns de Docker s’acustumen a guardar-se al directori /var/lib/docker/volumes del host
- Variables d’entorn
  - Contrasenya de root: fjeclot
  - Nom de la BD: wordpress
  - Nom d’usuari de MariaDB: manager
  - Contrasenya de l’usuari manager: fjeclot
- Imatge del contenidor
  - Última versió de MariaDB

```
sebagu@sebagu-asix:~$ docker run -d --name wordpress-db \
--mount source=wordpress-db,target=/var/lib/mysql \
-e MYSQL_ROOT_PASSWORD=fjeclot \
-e MYSQL_DATABASE=wordpress \
-e MYSQL_USER=manager \
-e MYSQL_PASSWORD=fjeclot mariadb:latest
```

Seguidament, amb l’ordre docker ps -a podrem veure que el contenidor s’ha creat correctament

```
sebagu@sebagu-asix:~$ docker ps -a
CONTAINER ID   IMAGE                COMMAND                  CREATED          STATUS          PORTS                NAMES
e2f02866a5c5   mariadb:latest       "docker-entrypoint.s…"   40 seconds ago   Up 39 seconds   3306/tcp             wordpress-db
```

Creació del contenidor amb Wordpress

El següent pas serà crear el contenidor amb Wordpress. Per això crearem la següent estructura, on el directori target serà on residiran tots els arxius de Wordpress.

```
sebagu@sebagu-asix: /var/ww
sebagu@sebagu-asix:/var/www/html/php/exercicis/practica7/7.1$ mkdir -p Sites/wordpress/target
```

```
sebagu@sebagu-asix:/var/www/html/php/exercicis/practica7/7.1$ tree
.
└── Sites
    └── wordpress
        └── target

4 directories, 0 files
sebagu@sebagu-asix:/var/www/html/php/exercicis/practica7/7.1$
```

Seguidament, crearem el contenidor amb les següents característiques:

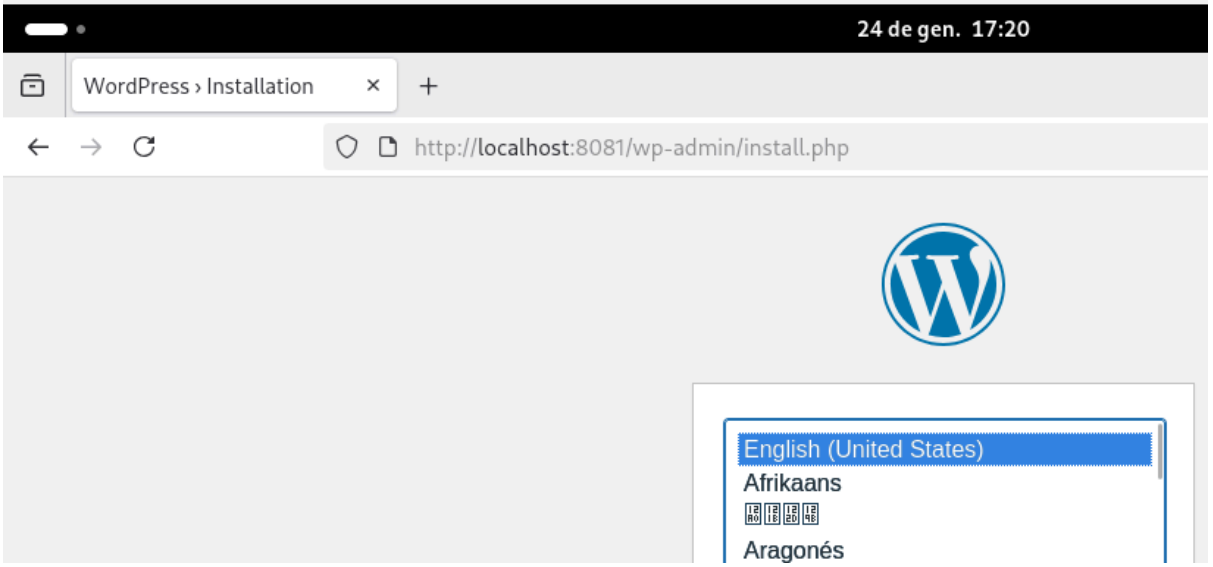
- Nom
  - El nom del contenidor serà wordpress
- Links
  - Al no fer servir Docker Compose, hem d'indicar que aquest Docker pot comunicar-se amb el Docker de MariaDB.
  - Farà servir el hostname "mysql" per comunicar-se amb el contenidor
  - Avui en dia s'hauria de fer servir una xarxa Docker enlloc d'aquest paràmetre
- Volums
  - Crearem un únic volum el qual serà un volum compartit entre el directori /var/www/html/php/exercicis/practica7/7.1/Sites/wordpress/target del host i el /var/www/html del Docker
  - En aquest cas especifiquem el tipus de volum "bind", indicant que el volum no és intern de Docker, sinó un directori del host
- Variables d'entorn
  - Usuari per accedir a la DB: manager
  - Contrasenya de l'usuari manager: fjeclot
- Ports exposats
  - El port 8081 del host redirigirà al port 80 del Docker
- Imatge del contenidor
  - Última versió de Wordpress

```
sebagu@sebagu-asix: /var/ww
sebagu@sebagu-asix:/var/www/html/php/exercicis/practica7/7.1/Sites/wordpress$ docker run -d --name wordpress \
--link wordpress-db:mysql \
--mount type=bind,source=/var/www/html/php/exercicis/practica7/7.1/Sites/wordpress/target,target=/var/www/html \
-e WORDPRESS_DB_USER=manager \
-e WORDPRESS_DB_PASSWORD=fjeclot \
-p 8081:80 \
wordpress:latest
```

Un cop creem el contenidor, amb l'ordre *docker ps -a* comprovarem que tenim 2 contenidors de Docker. El de la BD amb MariaDB i l'altre del propi Wordpress

```
sebagu@sebagu-asix: /var/ww
sebagu@sebagu-asix:/var/www/html/php/exercicis/practica7/7.1/Sites/wordpress$ docker ps -a | grep wordpress
7e7e821f522f   wordpress:latest              "docker-entrypoint.s…"   35 seconds ago   Up 33 seconds   0.0.0.0:8081->80/tcp, [::]:8081->80/tcp   wordpress
e2f02866a5c5   mariadb:latest                "docker-entrypoint.s…"   11 minutes ago   Up 10 minutes   3306/tcp                                  wordpress-db
sebagu@sebagu-asix:/var/www/html/php/exercicis/practica7/7.1/Sites/wordpress$
```

Si accedim a la màquina pel port 8081 veurem la pàgina inicial d'instal·lació de Wordpress.



Pràctica 7.2

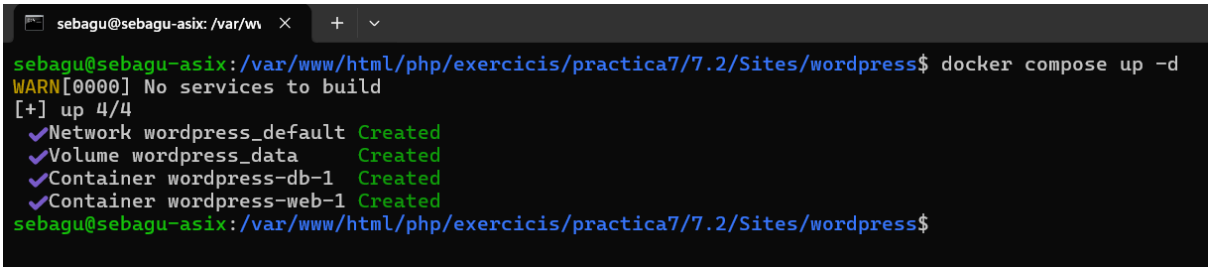
El que hem realitzat es pot simplificar molt més treballant amb Docker Compose. Per això farem un docker-compose.yaml amb les següents característiques:

1. Un contenidor anomenat db
    a. La imatge del contenidor serà la última versió de MariaDB
    b. Volums
        i. El contingut de /var/lib/mysql del Docker 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)
    c. Variables d'entorn
        i. La contrasenya de ROOT de MariaDB serà fjeclot
        ii. El nom de la BD serà wordpress
        iii. Un usuari de MariaDB s'anomenarà manager
        iv. La contrasenya de l'usuari manager serà fjeclot
2. Un contenidor anomenat web
    a. La imatge del contenidor serà la última versió de Wordpress
    b. Dependència del contenidor amb nom "db"
        i. Aquest contenidor no es posarà en marxa fins que el contenidor db no ho estigui
    c. Volums
        i. Els arxius del directori ./target del host es compartiran amb el directori /var/www/html del Docker
    d. Variables d'entorn
        i. L'usuari que es conectarà a la BD serà manager
        ii. La contrasenya de l'usuari manager serà fjeclot
        iii. El nom del host que allotja la BD serà "db" (nom del docker amb MariaDB)
        iv. El nom de la BD que es farà servir és wordpress
    e. Ports exposats
        i. El port 8082 del host dirigirà al port 80 del Docker

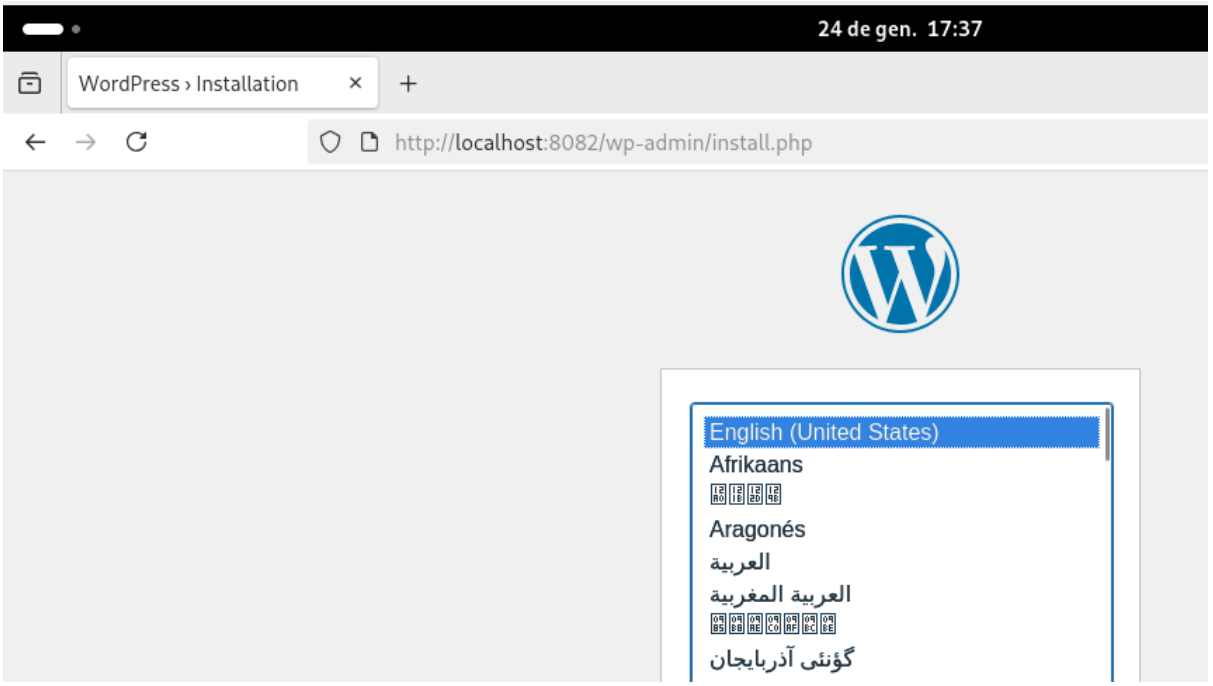
C: > Users > Usuario > Documents > PHP > exercicis > practica7 > 7.2 > Sites > wordpress > docker-compose.yml

```yaml
services:
  db:
    image: mariadb:latest # Fem servir la imatge oficial de mariadb, la última versió disponible
    volumes:
      - data:/var/lib/mysql # El contingut de /var/lib/mysql del Docker es desarà a un volum intern de Docker anomenat mysqldata (NO ÉS UN DIRECTORI DEL P
    environment:
      - MYSQL_ROOT_PASSWORD=secret # Definim la contrasenya de root de MariaDB
      - MYSQL_DATABASE=wordpress # Definim el nom de la BD
      - MYSQL_USER=manager # Definim un usuari de MariaDB
      - MYSQL_PASSWORD=fjeclot # Definim la contrasenya de l'usuari creat anteriorment
  web:
    image: wordpress:latest # Fem servir la imatge oficial de wordpress, la última versió disponible
    depends_on:
      - db # Requerim que el servei db estigui operatiu
    volumes:
      - ./target:/var/www/html # Definim un volum. Els documents del directori ./target del host es compartiran amb el directori /var/www/html del Docker
    environment:
      - WORDPRESS_DB_USER=manager # Definim l'usuari que farà servir Wordpress per la BD
      - WORDPRESS_DB_PASSWORD=fjeclot # Definim la contrasenya de l'usuari que farà servir Wordpress per la BD
      - WORDPRESS_DB_HOST=db # Definim el host on s'allotja la BD de Wordpress. Indiquem el nom del docker de MariaDB
      - WORDPRESS_DB_NAME=wordpress # Definim el nom de la BD que farem servir
    ports:
      - 8082:80 # El port 8082 del host dirigirà al port 80 del Docker

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

Per posar en marxa els serveis del Docker Compose, executarem l'ordre *docker compose up -d* en el directori on es troba el docker-compose.yaml



Si accedim a la màquina pel port 8082, veurem el mateix que hem vist anteriorment, la pàgina inicial d'instal·lació de Wordpress



Wordpress al VPS

Configuració de Dockers

Per executar Wordpress al VPS, farem servir el Docker Compose de la pràctica 7.2, amb algunes modificacions.

1. Un contenidor anomenat mariadb
    a. La imatge del contenidor serà la última versió de MariaDB
    b. Volums
        i. El contingut de /var/lib/mysql del Docker 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)
    c. Variables d'entorn
        i. La contrasenya de ROOT de MariaDB serà fjeclot
        ii. El nom de la BD serà wordpress
        iii. Un usuari de MariaDB s'anomenarà manager
        iv. La contrasenya de l'usuari manager serà fjeclot
2. Un contenidor anomenat wordpress
    a. La imatge del contenidor serà la última versió de Wordpress
    b. Dependència del contenidor amb nom "mariadb"
        i. Aquest contenidor no es posarà en marxa fins que el contenidor db no ho estigui
    c. Volums
        i. Els arxius del directori ./dadesWordpress del host es compartiran amb el directori /var/www/html del Docker
    d. Variables d'entorn
        i. L'usuari que es conectarà a la BD serà manager
        ii. La contrasenya de l'usuari manager serà fjeclot
        iii. El nom del host que allotja la BD serà "mariadb" (nom del docker amb MariaDB)
        iv. El nom de la BD que es farà servir és wordpress
    e. Ports exposats
        i. El port 8081 del host dirigirà al port 80 del Docker

Configuració del NSG d'Azure

Gràcies a la configuració del VPS que es va realitzar a la pràctica 6, no cal realitzar més accions sobre el NSG

Configuració de noms de domini amb Nominalia

Per aquesta pràctica únicament definiré un nom de domini: wordpress.gserrat.cat.

Per això, dins de Nominalia, el meu proveidor del domini, crearé els següents registres DNS:

- Registres A
  - wordpress.gserrat.cat -> IP del VPS
- Registres CNAME
  - www.wordpress.gserrat.cat -> wordpress.gserrat.cat

Configuració del host virtual 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
- ssl

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

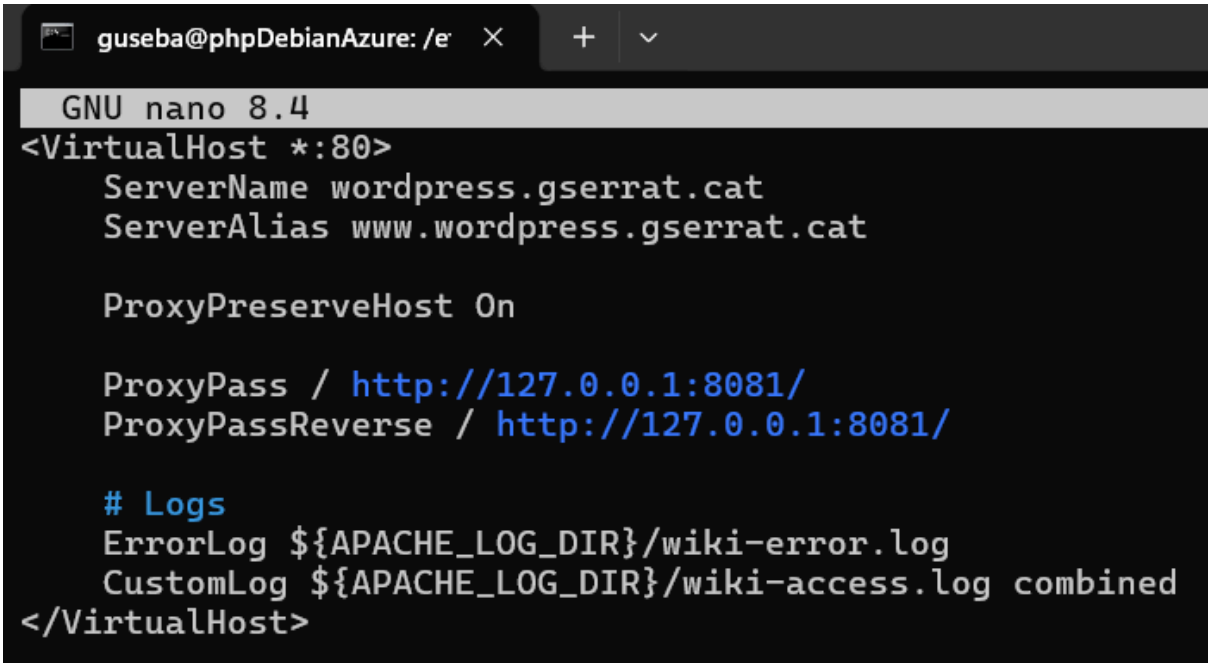
Configuració del host virtual

En el host virtual definirem el nom de domini wordpress.gserrat.cat i l'àlies www.wordpress.gserrat.cat.

En cas d'accedir al servidor amb algun d'aquests dos noms de domini, redirigirem a l'usuari a "http://127.0.0.1:8081", és a dir, al localhost al port 8081, on s'està exportant el docker amb Wordpress.

La "/" indica que accedim a l'arrel del directori, és a dir, wordpress.gserrat.cat i no wordpress.gserrat.cat/altre.html. En cas de fer-ho, redirigirà igualment al localhost pel port 8081

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.



Certificació SSL del host virtual

La certificació SSL ens permet encriptar el tràfic HTTP mitjançant HTTPS, fet imprescindible quan allotjem una pàgina web.

Primerament, haurem d'instal·lar les utilitats necessàries i preparar l'entorn:

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

Per certificar els dos noms de domini executarem la següent ordre *sudo certbot --apache -d wordpress.gserrat.cat -d www.wordpress.gserrat.cat*

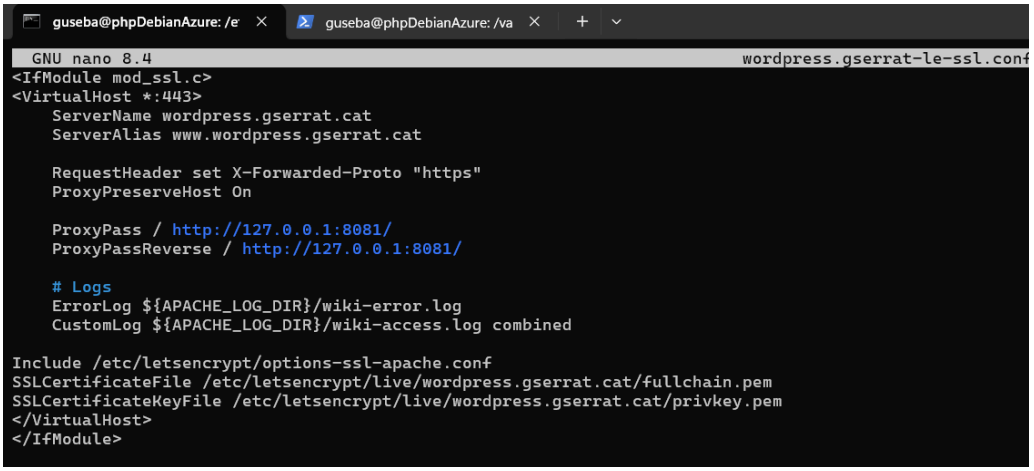
Un cop certificat ens generarà un nou host virtual que escoltarà pel port 443 d'Apache i on tindrà els mòduls de certificació en HTTPS.

És important modificar aquest host virtual i afegir la directiva *RequestHeader set X-Forwarded-Proto "https"*. Això és important un cop entenem el funcionament del proxy invers:

1. Quan el client es comunica amb el servidor fa servir HTTPS
2. El servidor rep la petició i actúa com a servidor intermediari
3. El servidor es comunica amb el Docker, però ho fa amb HTTP
4. El Docker respon la petició del servidor en HTTP
5. El servidor repon al client amb la informació del Docker però en HTTPS

Això, en cas de Wordpress és un problema, ja que si comprova que accedim mitjançant una URL amb HTTPS però la comunicació realment és HTTP, no funcionarà correctament

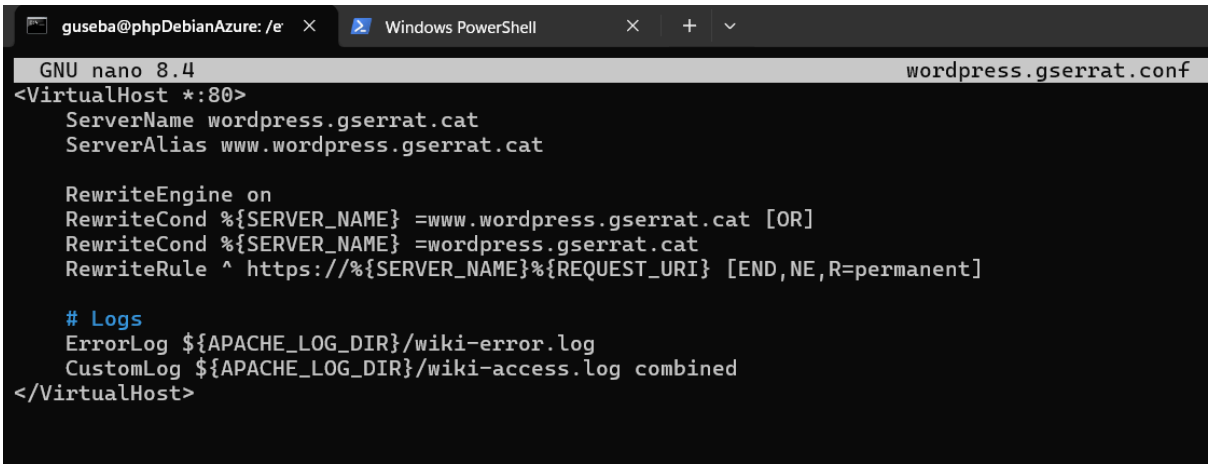
Amb la directiva anteriorment esmentada, el servidor modifica la capçalera HTTP per "fer veure" que la petició és HTTPS, tot i no estar certificada, per tal de que Wordpress conclogui que la connexió és HTTPS



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.



Instal·lació de Wordpress

Un cop tenim els contenidors en marxa, si accedim a la pàgina de Wordpress, podrem començar la seva instal·lació.

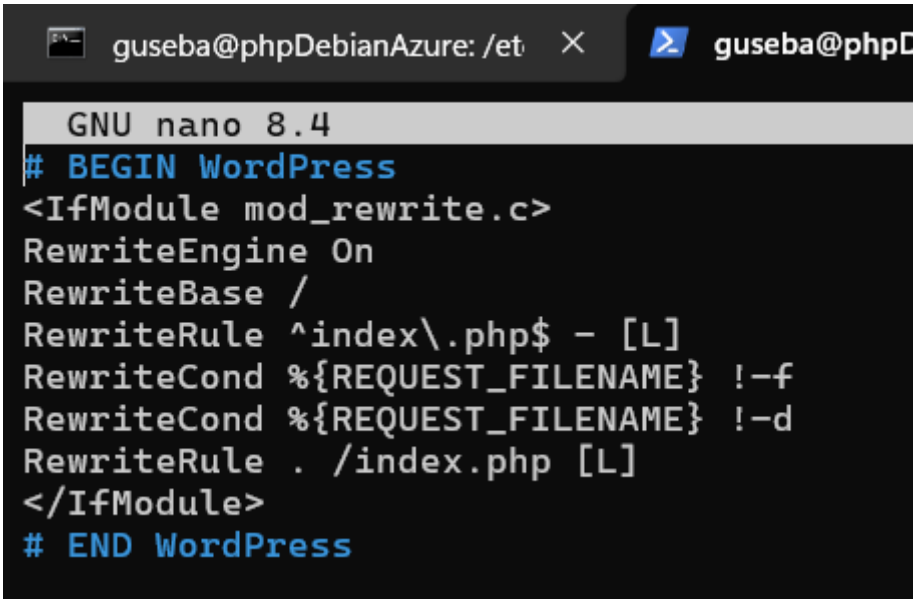
Tot i això és molt important realitzar algunes modificacions ens els fitxers de Wordpress per assegurar el bon funcionament amb HTTPS.

Primerament, modificarem el fitxer wp-config.php:

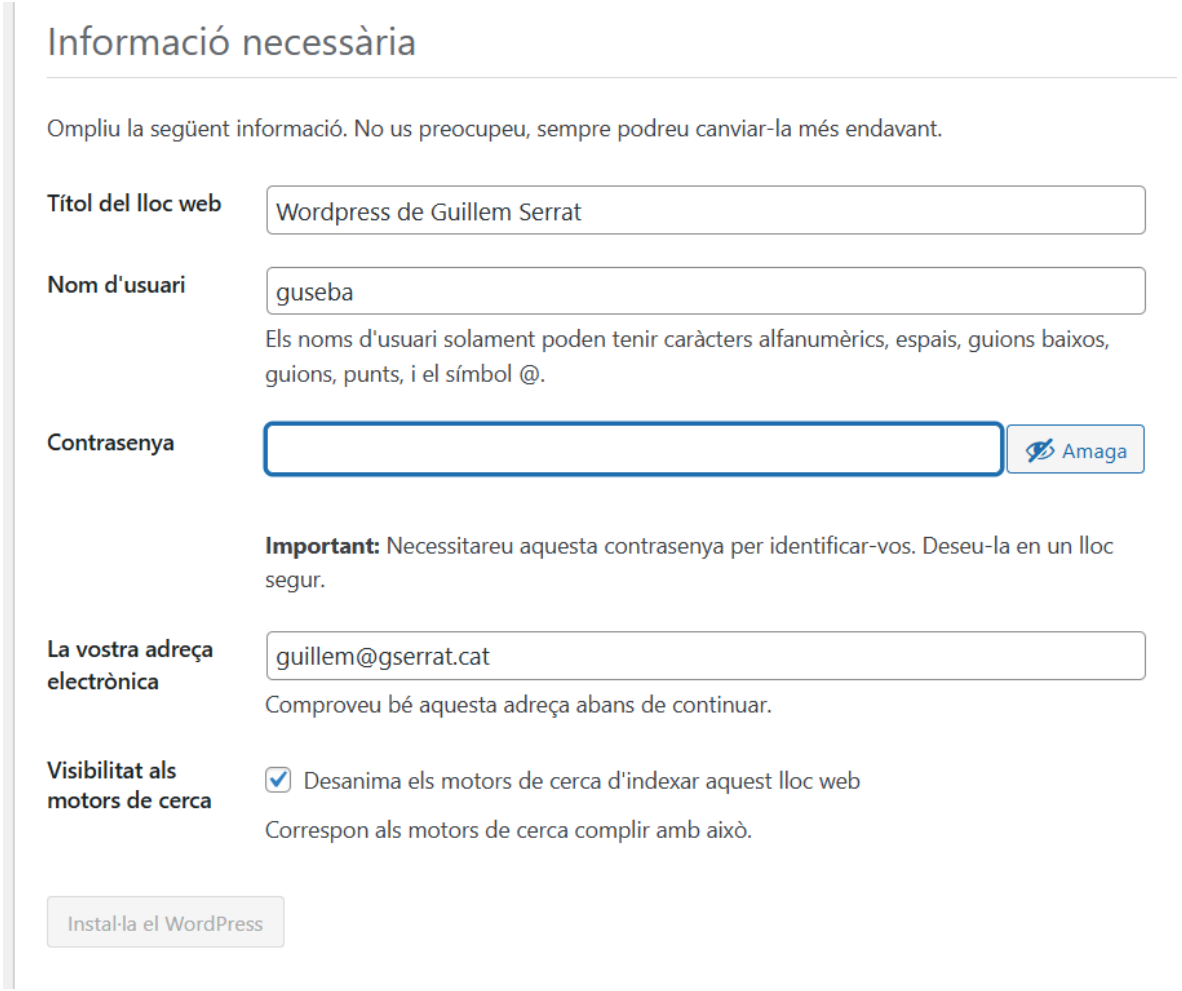
- Forçarem que el login d'administració sigui amb SSL
- Configurarem Wordpress per treballar sota un proxy, indicant que si la capçalera RequestHeader set X-Forwarded-Proto és "HTTPS" (indicat al host virtual), indicarem que el protocol HTTPS està habilitat, tot i que la comunicació interna Apache - Docker no ho sigui
- Definirem els enllaços de Wordpress amb HTTPS

```php
// Forzar HTTPS en el admin
define('FORCE_SSL_ADMIN', true);

// Corregir HTTPS detrás de un proxy
if (isset($_SERVER['HTTP_X_FORWARDED_PROTO']) && $_SERVER['HTTP_X_FORWARDED_PROTO'] === 'https') {
    $_SERVER['HTTPS'] = 'on';
}

// URL del sitio
define('WP_HOME', 'https://wordpress.gserrat.cat');
define('WP_SITEURL', 'https://wordpress.gserrat.cat');
```

A continuació modificarem el .htacces de la següent manera

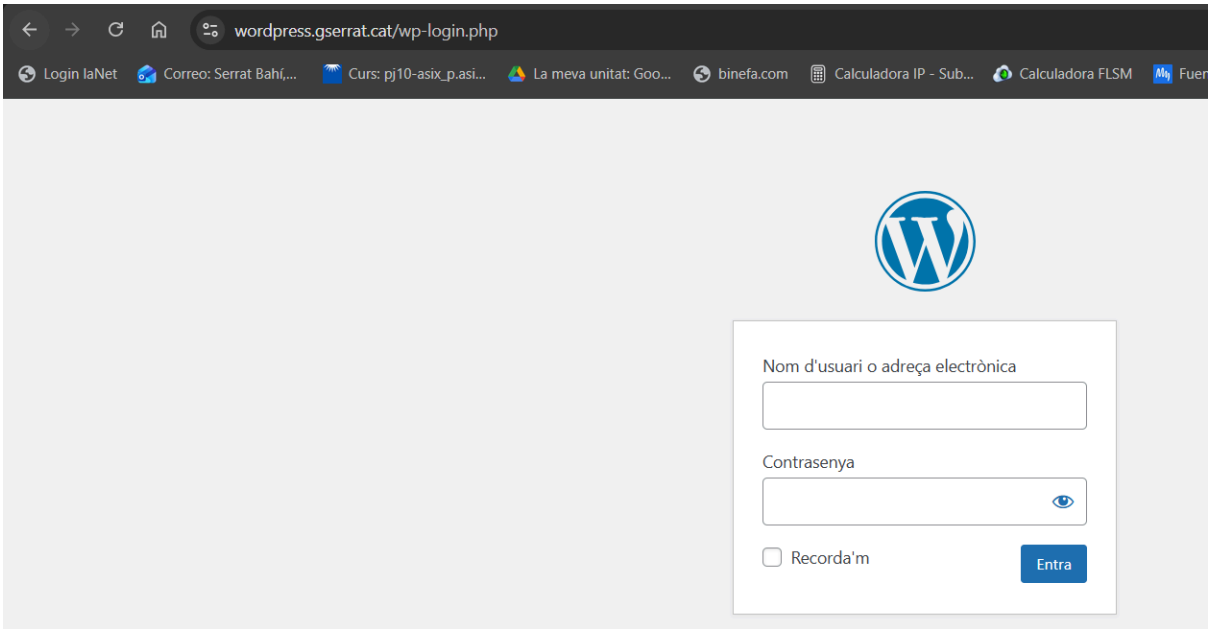


Un cop modificats els fitxers de configuració esmentats anteriorment, haurem d'accedir de nou a Wordpress (amb HTTPS), seleccionarem l'idioma i emplenarem la informació bàsica



Un cop emplenada, si tornem a entrar a Wordpress, veurem la pàgina inicial

En cas de voler iniciar sessió amb l'usuari creat anteriorment per poder editar articles, haurem de dirigir-nos a la URL del Wordpress amb la pàgina wp-login.php



Un cop hem iniciat sessió, veurem el tauler d'administració de Wordpress, des d'on podrem realitzar diferents tasques, una d'elles editar articles