

# Manual d'Administració VÍBRIA

## Tecnologies

- **WordPress** - CMS per al backend
- **Astro** - Generació de frontend estàtic
- **PhpMyAdmin** - Gestió de base de dades MySQL

## Accés

### Accés a WordPress

- URL: <http://localhost:8000/wp-admin>
- Usuari: **wordpress**
- Contrasenya: **wordpress**

### Accés a la Base de Dades (PhpMyAdmin)

- URL: <http://localhost:8080>
- Usuari: **wordpress**
- Contrasenya: **wordpress**

## Connexió entre Backend i Frontend

- **Astro** consumeix contingut de **WordPress** mitjançant la seva REST API.
- El contingut es mostra en el frontend renderitzat de manera estàtica.

## Estructura del Frontend (Astro)

### Comportament de les Rutes

Ruta	Funció
/	Redirecció a la home
/ca/home-ca	Home en català
/es/home-es	Home en castellà
/ca/ofertes-ca	Llistat d'ofertes
/ca/ofertes-ca/oferta-italia-ca	Detall d'una oferta

## Comandes de Desenvolupament

Comanda	Funció
<code>npm install</code>	Instal·la les dependències del projecte
<code>npm run build</code>	Genera els arxius per a producció
<code>npm run preview</code>	Previsualitza el build generat

## Plugins Instal·lats

Plugin	Funció
Advanced Custom Fields (ACF)	Camps personalitzats per gestionar continguts complexos
Code Snippets	Afegir fragments de codi PHP sense editar fitxers
Custom Post Type UI (CPT)	Crear tipus de contingut personalitzats
Polylang	Traducció i gestió multilingüe del contingut
WP Mail SMTP	Configuració d'enviament de correus mitjançant SMTP

## Respatller i Restauració de la Base de Dades

### Actualitzar Docker amb els canvis del repositori Git

Si has fet canvis al repositori i necessites actualitzar la base de dades a Docker, hauries d'estar en la carpeta server i si estàs en Windows fer-lo en el Git bash, utilitza:

```
cd server
docker exec -i server-db-1 mysql --default-character-set=utf8mb4 -u wordpress
-pwordpress wordpress < ../data/backup.sql
```

### Guardar els canvis per pujar-los a Git

Per fer una còpia de seguretat de la base de dades i pujar-la al repositori, hauries d'estar en la carpeta server i si estàs en Windows fer-lo en el Git bash.:

```
cd server
docker exec server-db-1 mysqldump --default-character-set=utf8mb4 -u wordpress
-pwordpress wordpress > ../data/backup.sql
```

# Actualitzar Docker amb les imatges del repositori

Després de carregar la base de dades, pots recuperar les imatges que ha pujat el teu company amb:

```
docker cp ../Blog/wordpress/wp-content/uploads/2025/05 server-wordpress-1:/var/www/html/wp-content/uploads/2025/
```

## Erros comuns i solucions

### Si no pots guardar els canvis: accedir a la base de dades

Si obtens un error de permisos en fer el backup, accedeix primer a la base de dades:

```
docker exec -it server-db-1 mysql -u root -p
```

### Donar permisos al usuari de WordPress

Després, atorga els permisos necessaris a l'usuari de WordPress perquè pugui realitzar les accions necessàries:

```
GRANT PROCESS, LOCK TABLES, SELECT, RELOAD, SHOW DATABASES, EVENT, TRIGGER, SHOW VIEW  
ON *.* TO 'wordpress'@'%';  
FLUSH PRIVILEGES;
```

Torna a intentar guardar els canvis amb:

```
docker exec server-db-1 mysqldump -u wordpress -pwordpress wordpress >  
../data/backup.sql
```

### Si WordPress demana permisos per instal·lar un plugin

Per solucionar problemes de permisos en instal·lar plugins:

```
docker exec -it server-wordpress-1 chown -R www-data:www-data /var/www/html
```

# Canvi de configuració de codificació a la base de dades

## Accedir a la base de dades

```
docker exec -it server-db-1 mysql -u root -p
```

## Modificar la codificació i col·lació

```
ALTER DATABASE wordpress CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;
```

## Verificar que el canvi s'ha aplicat correctament

```
SELECT schema_name, default_character_set_name, default_collation_name
FROM information_schema.schemata
WHERE schema_name = 'wordpress';
```

Sortida esperada:

schema_name	default_character_set_name	default_collation_name
wordpress	utf8mb4	utf8mb4_unicode_ci

Ara WordPress hauria de gestionar correctament els accents i caràcters especials.

## Si WordPress mostra error crític després de pujar tot

Això pot ser perquè no s'han instal·lat bé els plugins. Prova el següent:

```
# 1. Eliminar vendor i lock file
cd Blog/wordpress
Remove-Item -Force composer.lock, vendor -Recurse

# 2. Reinstal·lar tot
composer install --optimize-autoloader
```

## Desplegament del Frontend

```
cd vibria-brian-oscar-adrian/astro
npm install
npm run build
```

# Manual AWS

## Crear una instància EC2

### Accedir a la consola de gestió

Un cop hagiis iniciat sessió, accedeix a la AWS Academy Learner Lab. Un cop allà, Vas a "Contenidos" > "Lanzamiento del Laboratorio para el alumnado de AWS Academy". Un cop ho iniciem clickant "Start Lab", ens apareixerà AWS amb una icone en verd, hi clickem i ens redirigeix a la "Página de inicio de la consola".

Li donem a l'opció de "AWS Details" i a l'apartat de SSH Key premem a Download PPK (en cas de Windows) o Download PEM (en cas de Linux o Mac) per connectar-ho per SSH.

### Creació de l'instància EC2

Un cop hem clickat a AWS quan està verd, ens redirigirà a la "Página de inicio de la consola", i llavors premem l'opció de EC2 i fem scroll fins l'apartat on posa "Lanzar la instancia" i llavors començem amb configuració de la màquina.

Hi posem el nom, a "Imágenes de máquina de Amazon" escollim la que vulguem, en el nostre cas, escollim "Ubuntu Server 20.04 LTS (HVM), SSD Volume Type", i li donem a "Seleccionar".

Després, a "Seleccionar tipus d'instància" escollim la que vulguem, en el nostre cas, escollim "t2.micro" i a "Par de claves (inicio de sesión)", escollim l'opció per defecte de vockey. A continuació, a "Configuraciones de red", premem els checkboks de "Permitir el tráfico de HTTPS desde Internet" i de "Permitir el tráfico de HTTP desde Internet".

Per últim, configurem l'emmagatzemmatge amb 30 GiB i un cop fet tot això, li donem a "Lanzar la instancia".

### Connexió SSH a la instància EC2 (Linux/ubuntu)

Podem connectar-nos a la instància EC2 mitjançant SSH. Per fer-ho, podem utilitzar el programari PuTTY (en cas de Windows) o la terminal mateixa amb OpenSSH.

### Descarreguem PuTTY en local i ens connectem a la instància EC2 (Windows)

Descarreguem el software PuTTY a Windows, per tal de poder connectar-nos a la nostre instància EC2. Amb la clau privada (.ppk) que hem descarregat anteriorment, obrim PuTTYgen, "File" > "Load Private Key", escollim la .ppk, i li donem a l'opció de "Save private key".

Ara obrim PuTTY i configurem la connexió SSH amb la nostra instància EC2. Introduïm l'adreça IP de la instància (en el nostre cas, 44.210.150.195) a la casella "Host Name (or IP address)" i ens

assegurem de que el port sigui el 22. A la secció "Connection" > "SSH" > "Auth" > "Credencials", on posa "Private key file for authentication" seleccionem el fitxer .ppk que hem creat amb PuTTYgen. Finalment, fes clic a "Open" per connectar-te a la instància.

Per agilitza-ho, podem guardar la configuració de la connexió al subapartat "Saved Sessions" per tal de no haver de configurar-la cada vegada que ens connectem a la instància.

Finalment, després de donar-li a "Open", ens hauria de sortir una finestra de connexió a la instància EC2. Ens demanarà un usuari, en el nostre cas, l'usuari és "ubuntu".

Un cop ens haguem connectat a la instància, ja podrem començar a treballar amb ella.

## Amb OpenSSH(Windows)

Simplement hem d'anar a EC2 > Instancias > ID de la instancia > Conectarse a la instancia i llavors a la finestra de "Cliente SSH".

Allà on posa "Ejemplo:", copiem la comanda, com per exemple la següent: `ssh -i "vockey.pem" ubuntu@ec2-54-83-183-202.compute-1.amazonaws.com` i obrim la terminal amb drets d'administrador. Ens movem a la carpeta on tenim el fitxer .PEM, i només hem de canviar el nom del fitxer per defecte de "vockey.pem" per el nostre i executar la comanda.

## Instal·lació del web server a AWS EC2

- 1. Actualitzem el sistema amb la comanda:

```
sudo apt update -y
```

```
sudo apt upgrade -y
```

- 2. Instal·lem el servidor web Apache amb la comanda:

```
sudo apt install apache2 -y
```

- 3. Per tal de que el servei apache pugui permetre reescriure les URL, cal activar el mòdul de reescriptura amb la comanda:

```
sudo a2enmod rewrite
```

- 4. Per tal de que el servei apache s'habiliti automàticament en iniciar el sistema, cal activar el servei amb la comanda:

```
sudo systemctl restart apache2
```

- 5. Creem l'arxiu de configuració de vibria:

```
cd /etc/apache2/sites-available/  
sudo nano vibria.conf
```

- 6. Afegim el següent contingut a l'arxiu de configuració amb el seu directori arrel i els certificats d'SSL:

```
<VirtualHost *:80>  
    ServerName wordpress.vibriaoba.store  
    ServerAdmin webmaster@vibriaoba.store  
    DocumentRoot /var/www/html  
  
    ErrorLog ${APACHE_LOG_DIR}/wordpress_error.log  
    CustomLog ${APACHE_LOG_DIR}/wordpress_access.log combined  
  
    #RewriteEngine On  
    #RewriteCond %{HTTPS} off  
    #RewriteRule ^ https://%{HTTP_HOST}%{REQUEST_URI} [L,R=301]  
  
    ProxyPreserveHost On  
    ProxyPass / http://localhost:8000/  
    ProxyPassReverse / http://localhost:8000/  
  
</VirtualHost>  
  
<VirtualHost *:80>  
    ServerName vibria.vibriaoba.store  
    DocumentRoot /var/www/html/vibria-brian-oscar-adrian/astro/dist/client/  
    ErrorLog ${APACHE_LOG_DIR}/vibria_error.log  
    CustomLog ${APACHE_LOG_DIR}/vibria_access.log combined  
  
    # Redirige todo a HTTPS  
    # RewriteEngine On  
    # RewriteCond %{HTTPS} off  
    # RewriteRule ^(.*)$ https://%{HTTP_HOST}%{REQUEST_URI} [L,R=301]  
</VirtualHost>
```



Si voleu podreu separar la part de la vibria (astro) en un servidor i el WordPress a altre servidor només hauríeu de canviar el .env i el Cloudflare para que apunti a la nova IP.

```
WP_DOMAIN="https://EL_TEU_DNS"
```

- 7. Desabilitem l'arxiu de configuració de 000-default.conf amb la comanda:

```
sudo a2dissite 000-default.conf
```

- 8. Habilitem l'arxiu de configuració de vibria.conf amb la comanda:

```
sudo a2ensite vibria.conf
```

- 9. Habilitem el mòdul proxy i proxy\_http d'Apache.

```
sudo a2enmod proxy proxy_http
```

- 10. Reiniciem el servei apache amb la comanda:

```
sudo systemctl restart apache2
```

- 11. Donem els permisos al Directori on posarem el nostre projecte

```
sudo chown -R www-data:www-data /var/www/html  
sudo chmod -R 755 /var/www/html
```

## Instal·lació de Git a AWS EC2

- 1. Instal·lem Git amb la comanda:

```
sudo apt install git -y
```

- 2. Clonem el repositori del nostre projecte amb les següents comandes:

```
cd /var/www/html/  
  
sudo git clone https://gitlab.com/abernabeu1/vibria-brian-oscar-adrian.git
```

- 3. Un cop hem creat el nostre projecte, canviem l'arxiu .env.example a .env per així configurar l'entorn d'aplicació:

```
cd /var/www/html/vibria-brian-oscar-adrian/astro  
sudo cp .env.example .env
```

- 4. Ara revisarem el .env i si no té aquesta línia d'aquesta forma ho canviem.



```
sudo nano .env
```

```
WP_DOMAIN="https://wordpress.vibriaoba.store"
```

## Instal·lació del Docker AWS EC2

- 1. Primer, amb la següent comanda desinstalem totes les versions antigues de Docker:

```
for pkg in docker.io docker-doc docker-compose docker-compose-v2 podman-docker  
containerd runc; do sudo apt-get remove $pkg; done
```

Ara si, instal·lem docker

- 2. Actualitzem el sistema amb la comanda:

```
sudo apt update -y
```

- 3. Afegim la clau GPG oficial de Docker:

```
sudo apt install -y ca-certificates curl  
  
sudo install -m 0755 -d /etc/apt/keyrings  
  
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o  
/etc/apt/keyrings/docker.asc  
  
sudo chmod a+r /etc/apt/keyrings/docker.asc
```

- 4. Afegim el repositori a les fonts d'APT:

```
echo \  
"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc]  
https://download.docker.com/linux/ubuntu \  
$(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \  
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

- 5. Actualitzem el la clau i el repositori

```
sudo apt update
```

- 6. Instalem el docker

```
sudo apt install -y docker-ce docker-ce-cli containerd.io \
docker-buildx-plugin docker-compose-plugin
```

- 7. Creem el grup docker

```
sudo groupadd docker
```

- 8. Afegim el teu usuari al grup Docker

```
sudo usermod -aG docker $USER
```

- 9. Apliquem els canvis amb aquesta comanda

```
newgrp docker
```

- 10. Puguem el docker amb la BBDD amb la comanda:

```
cd /var/www/html/vibria-brian-oscar-adrian/server
docker compose up -d
```

## Afegir la bd al docker

- 1. Verifiquem la codificació de l'arxiu SQL

```
docker exec -i server-db-1 mysql -u wordpress -pwordpress wordpress <
../data/backup.sql
```

- 2. Actualitzem els URL crítics

```
docker exec -i server-db-1 mysql -u root -ppassword wordpress <<'EOF'
SET FOREIGN_KEY_CHECKS=0;

UPDATE wp_options SET
  option_value = 'https://wordpress.vibriaoba.store'
WHERE option_name IN ('siteurl', 'home');

UPDATE wp_posts SET
  guid = REPLACE(guid, 'http://localhost:8000', 'https://wordpress.vibriaoba.store'),
  post_content = REPLACE(post_content, 'http://localhost:8000',
'https://wordpress.vibriaoba.store');

UPDATE wp_postmeta SET
```

```
meta_value = REPLACE(meta_value, 'http://localhost:8000',  
'https://wordpress.vibriaoba.store');
```

```
SET FOREIGN_KEY_CHECKS=1;  
EOF
```

- 3. Verifiquem les taules importades

```
docker exec -it server-db-1 mysql -u root -ppassword -e "SHOW TABLES;" wordpress
```

- 4. Verifiquem els URL actualitzats

```
docker exec -it server-db-1 mysql -u root -ppassword -e "  
  SELECT option_name, option_value  
  FROM wp_options  
  WHERE option_name IN ('siteurl', 'home');  
" wordpress
```

## 3.6 Instal·lació del Composer a AWS EC2

- 1. Instal·lació i actualització del composer i les dependències pertinents amb les següents comandes:

```
sudo apt install composer -y
```

- 2. Borrem el vendor y el arxiu composer.lock

```
cd /var/www/html/vibria-brian-oscar-adrian/Blog/wordpress/  
sudo rm -rf composer.lock vendor/
```

- 3. Tornem a instal·lar tot amb les seves dependències i optimitzem am els autoloader de les classes.

```
sudo composer install --optimize-autoloader
```

## Instal·lació de l'npm a AWS EC2

- 1. Instal·lem NPM amb la comanda:

```
sudo apt install npm -y
```

Si teniu el servidor a AWS, teniu poca RAM i es bloqueja sovint, feu això per afegir swap:



```
sudo fallocate -l 2G /swapfile
sudo chmod 600 /swapfile
sudo mkswap /swapfile
sudo swapon /swapfile
```

- 2. Executem la següent comanda per tal d'instal·lar les dependències del nostre projecte:

```
cd /var/www/html/vibria-brian-oscar-adrian/astro/
sudo npm install
```

- 3. Compilem les dades de Vite a la aplicació amb la comanda:

```
sudo npm run build
```

## Manual a seguir després de cada git pull

- Part del WordPress



```
cd /var/www/html/vibria-brian-oscar-adrian/server
sudo docker compose down -v
sudo docker compose up -d

docker exec -i server-db-1 mysql -u wordpress -pwordpress wordpress <
../data/backup.sql

docker exec -i server-db-1 mysql -u root -ppassword wordpress <<'EOF'
SET FOREIGN_KEY_CHECKS=0;

UPDATE wp_options SET
  option_value = 'https://wordpress.vibriaoba.store'
WHERE option_name IN ('siteurl', 'home');

UPDATE wp_posts SET
  guid = REPLACE(guid, 'http://localhost:8000',
'https://wordpress.vibriaoba.store'),
  post_content = REPLACE(post_content, 'http://localhost:8000',
'https://wordpress.vibriaoba.store');

UPDATE wp_postmeta SET
  meta_value = REPLACE(meta_value, 'http://localhost:8000',
'https://wordpress.vibriaoba.store');
```

```
SET FOREIGN_KEY_CHECKS=1;  
EOF
```

- Part del Astro

```
cd /var/www/html/vibria-brian-oscar-adrian/astro  
sudo run build
```