27/11/24

IMPLANTACIÓN DE APLICACIONES WEB

Jose Materano

Actividad 2 Manual de Despliegue de WordPress y Drupal en Docker

ITC Cuatrovientos - Pamplona

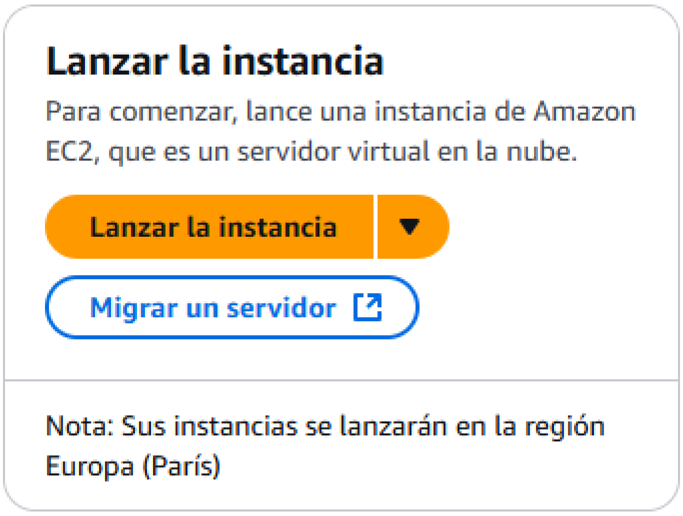
TABLA DE CONTENIDOS

Act 2. Manual de Despliegue de WordPress y Drupal en Docker

2. Requisitos Previos

Este documento detalla el paso a paso para el despliegue de los CMS WordPress y Drupal utilizando Docker. Se incluyen los comandos necesarios, explicaciones de cada paso todo desde un entorno AWS

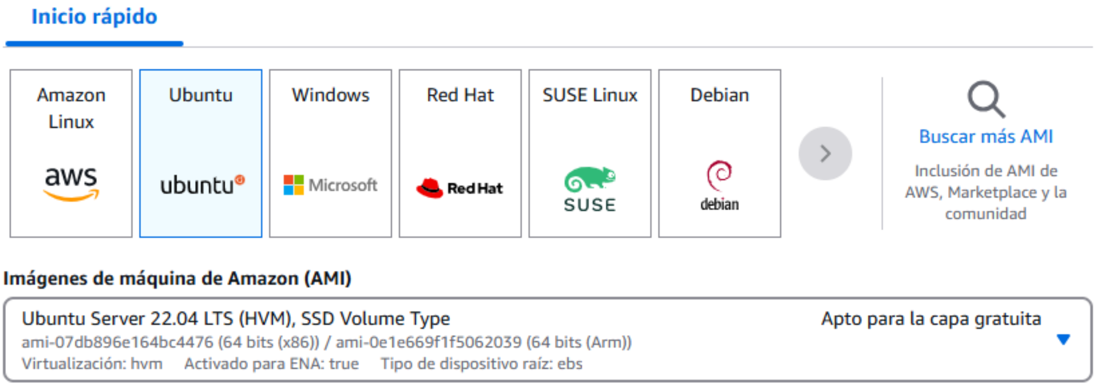
- Sistema Operativo: Ubuntu Server



Lanzar la instancia

Para comenzar, lance una instancia de Amazon EC2, que es un servidor virtual en la nube.

Nota: Sus instancias se lanzarán en la región Europa (París)



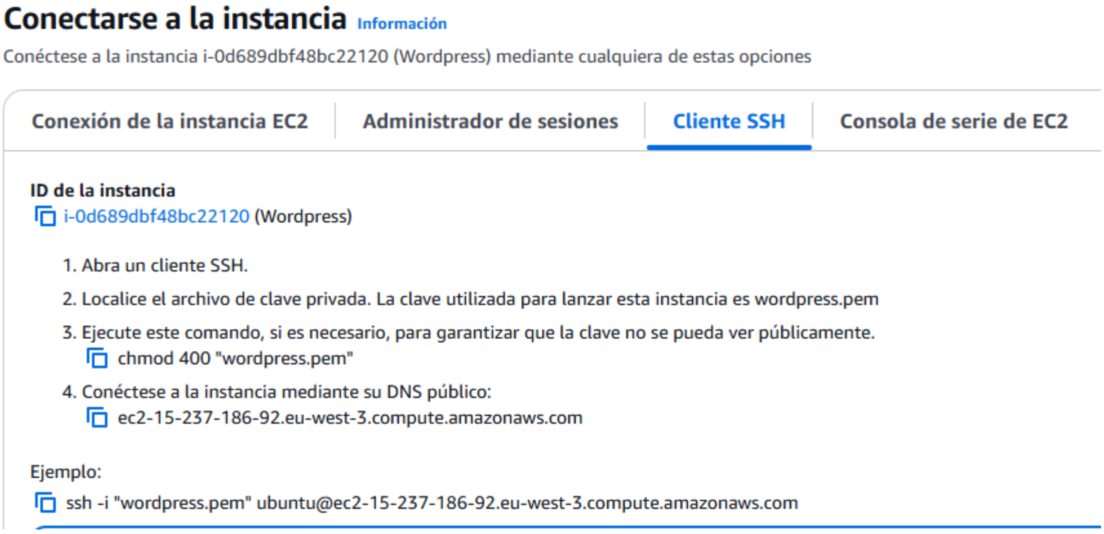
Imágenes de máquina de Amazon (AMI)

Ubuntu Server 22.04 LTS (HVM), SSD Volume Type — Apto para la capa gratuita

ami-07db896e164bc4476 (64 bits (x86)) / ami-0e1e669f1f5062039 (64 bits (Arm))

Virtualización: hvm Activado para ENA: true Tipo de dispositivo raíz: ebs

- Conexión por ssh a nuestra maquina



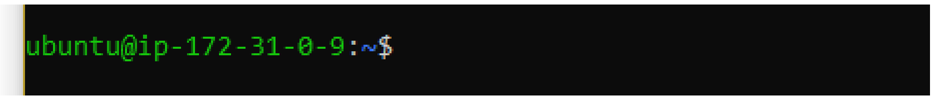
Conectarse a la instancia Información

Conéctese a la instancia i-0d689dbf48bc22120 (Wordpress) mediante cualquiera de estas opciones

ID de la instancia
i-0d689dbf48bc22120 (Wordpress)

1. Abra un cliente SSH.
2. Localice el archivo de clave privada. La clave utilizada para lanzar esta instancia es wordpress.pem
3. Ejecute este comando, si es necesario, para garantizar que la clave no se pueda ver públicamente.
   chmod 400 "wordpress.pem"
4. Conéctese a la instancia mediante su DNS público:
   ec2-15-237-186-92.eu-west-3.compute.amazonaws.com

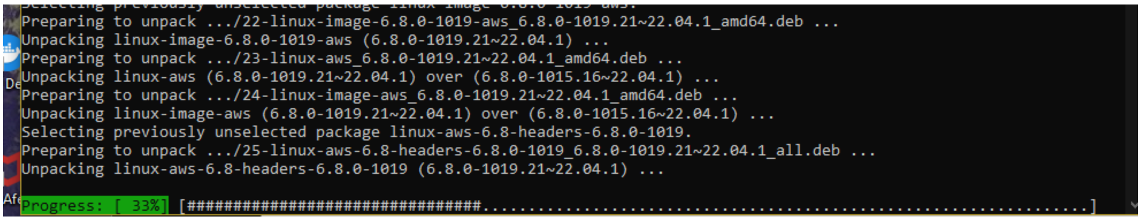
Ejemplo:
ssh -i "wordpress.pem" ubuntu@ec2-15-237-186-92.eu-west-3.compute.amazonaws.com



```
ubuntu@ip-172-31-0-9:~$
```

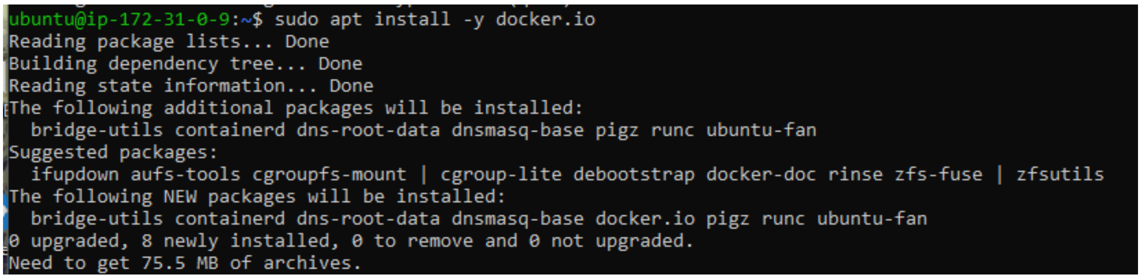
3. Instalar Docker

- sudo apt update y sudo apt upgrade



```
Selecting previously unselected package linux-image-6.8.0-1019-aws.
Preparing to unpack .../22-linux-image-6.8.0-1019-aws_6.8.0-1019.21~22.04.1_amd64.deb ...
Unpacking linux-image-6.8.0-1019-aws (6.8.0-1019.21~22.04.1) ...
Preparing to unpack .../23-linux-aws_6.8.0-1019.21~22.04.1_amd64.deb ...
Unpacking linux-aws (6.8.0-1019.21~22.04.1) over (6.8.0-1015.16~22.04.1) ...
Preparing to unpack .../24-linux-image-aws_6.8.0-1019.21~22.04.1_amd64.deb ...
Unpacking linux-image-aws (6.8.0-1019.21~22.04.1) over (6.8.0-1015.16~22.04.1) ...
Selecting previously unselected package linux-aws-6.8-headers-6.8.0-1019.
Preparing to unpack .../25-linux-aws-6.8-headers-6.8.0-1019_6.8.0-1019.21~22.04.1_all.deb ...
Unpacking linux-aws-6.8-headers-6.8.0-1019 (6.8.0-1019.21~22.04.1) ...

Progress: [ 33%] [################################..................................................................]
```

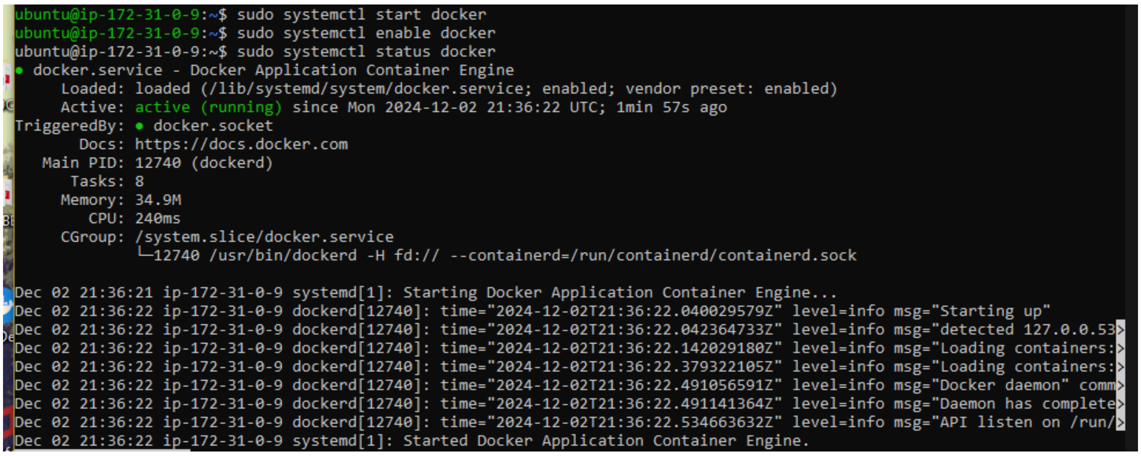
- sudo apt install -y docker.io



```
ubuntu@ip-172-31-0-9:~$ sudo apt install -y docker.io
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base pigz runc ubuntu-fan
Suggested packages:
  ifupdown aufs-tools cgroupfs-mount | cgroup-lite debootstrap docker-doc rinse zfs-fuse | zfsutils
The following NEW packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base docker.io pigz runc ubuntu-fan
0 upgraded, 8 newly installed, 0 to remove and 0 not upgraded.
Need to get 75.5 MB of archives.
```

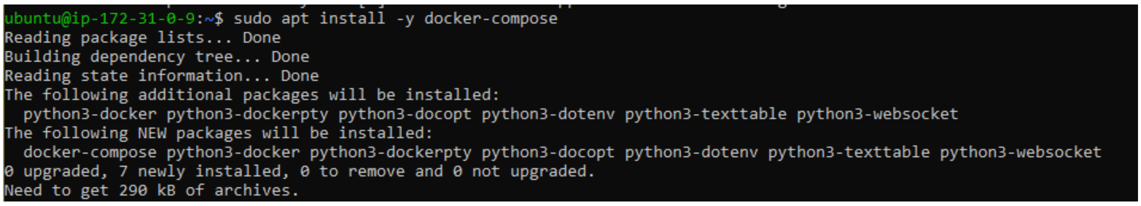
- sudo systemctl start docker
- sudo systemctl enable docker
- sudo systemctl status Docker

  Todo esto para verificar que el Docker se haya instalado correctamente



```
ubuntu@ip-172-31-0-9:~$ sudo systemctl start docker
ubuntu@ip-172-31-0-9:~$ sudo systemctl enable docker
ubuntu@ip-172-31-0-9:~$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
     Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
     Active: active (running) since Mon 2024-12-02 21:36:22 UTC; 1min 57s ago
TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
   Main PID: 12740 (dockerd)
      Tasks: 8
     Memory: 34.9M
        CPU: 240ms
     CGroup: /system.slice/docker.service
             └─12740 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

Dec 02 21:36:21 ip-172-31-0-9 systemd[1]: Starting Docker Application Container Engine...
Dec 02 21:36:22 ip-172-31-0-9 dockerd[12740]: time="2024-12-02T21:36:22.040029579Z" level=info msg="Starting up"
Dec 02 21:36:22 ip-172-31-0-9 dockerd[12740]: time="2024-12-02T21:36:22.042364733Z" level=info msg="detected 127.0.0.53>
Dec 02 21:36:22 ip-172-31-0-9 dockerd[12740]: time="2024-12-02T21:36:22.142029180Z" level=info msg="Loading containers:>
Dec 02 21:36:22 ip-172-31-0-9 dockerd[12740]: time="2024-12-02T21:36:22.379322105Z" level=info msg="Loading containers:>
Dec 02 21:36:22 ip-172-31-0-9 dockerd[12740]: time="2024-12-02T21:36:22.491056591Z" level=info msg="Docker daemon" comm>
Dec 02 21:36:22 ip-172-31-0-9 dockerd[12740]: time="2024-12-02T21:36:22.491141364Z" level=info msg="Daemon has complete>
Dec 02 21:36:22 ip-172-31-0-9 dockerd[12740]: time="2024-12-02T21:36:22.534663632Z" level=info msg="API listen on /run/>
Dec 02 21:36:22 ip-172-31-0-9 systemd[1]: Started Docker Application Container Engine.
```

- Docker Compose (opcional pero recomendado): Para manejar los contenedores de manera más sencilla:

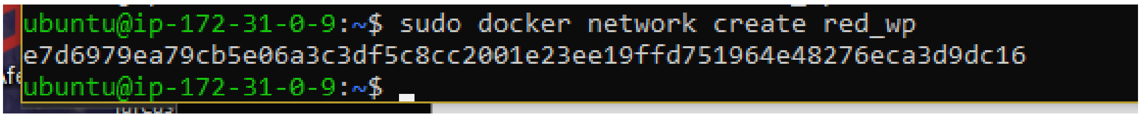


```
ubuntu@ip-172-31-0-9:~$ sudo apt install -y docker-compose
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  python3-docker python3-dockerpty python3-docopt python3-dotenv python3-texttable python3-websocket
The following NEW packages will be installed:
  docker-compose python3-docker python3-dockerpty python3-docopt python3-dotenv python3-texttable python3-websocket
0 upgraded, 7 newly installed, 0 to remove and 0 not upgraded.
Need to get 290 kB of archives.
```

4. Crear la Red red_wp

Este comando crea una red en la que ambos contenedores (WordPress y MySQL) estarán conectados:

docker network create red_wp



```
ubuntu@ip-172-31-0-9:~$ sudo docker network create red_wp
e7d6979ea79cb5e06a3c3df5c8cc2001e23ee19ffd751964e48276eca3d9dc16
ubuntu@ip-172-31-0-9:~$ 
```

5. Desplegar el Contenedor de MySQL

Ejecuta el siguiente comando para desplegar el contenedor con el servidor de base de datos MariaDB:

docker run -d --name servidor_mysql \

--network red_wp \

-v /opt/mysql_wp:/var/lib/mysql \

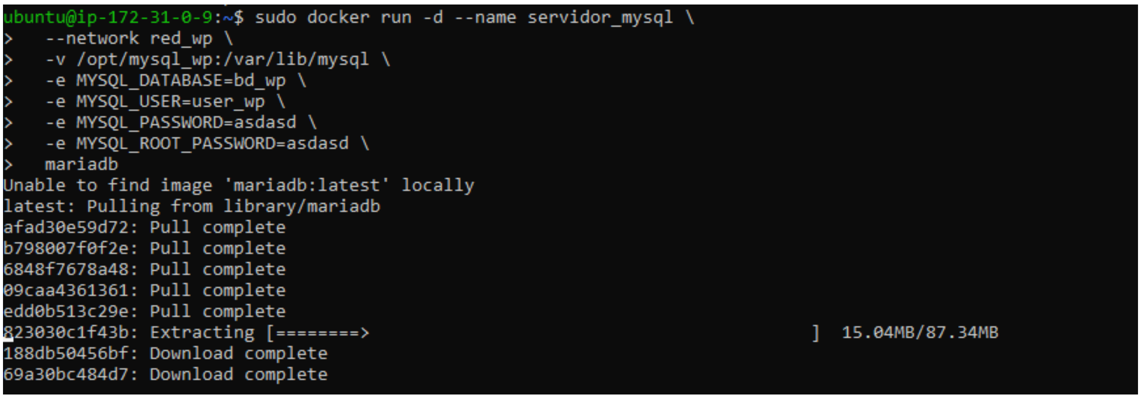
-e MYSQL_DATABASE=bd_wp \

-e MYSQL_USER=user_wp \

-e MYSQL_PASSWORD=asdasd \

-e MYSQL_ROOT_PASSWORD=asdasd \

mariadb



```
ubuntu@ip-172-31-0-9:~$ sudo docker run -d --name servidor_mysql \
>   --network red_wp \
>   -v /opt/mysql_wp:/var/lib/mysql \
>   -e MYSQL_DATABASE=bd_wp \
>   -e MYSQL_USER=user_wp \
>   -e MYSQL_PASSWORD=asdasd \
>   -e MYSQL_ROOT_PASSWORD=asdasd \
>   mariadb
Unable to find image 'mariadb:latest' locally
latest: Pulling from library/mariadb
afad30e59d72: Pull complete
b798007f0f2e: Pull complete
6848f7678a48: Pull complete
09caa4361361: Pull complete
edd0b513c29e: Pull complete
823030c1f43b: Extracting [========>                                          ]  15.04MB/87.34MB
188db50456bf: Download complete
69a30bc484d7: Download complete
```

6. Desplegar el Contenedor de WordPress

Ejecuta el siguiente comando para desplegar el contenedor WordPress:

docker run -d --name servidor_wp \

--network red_wp \

-v /opt/wordpress:/var/www/html/wp-content \

-e WORDPRESS_DB_HOST=servidor_mysql \

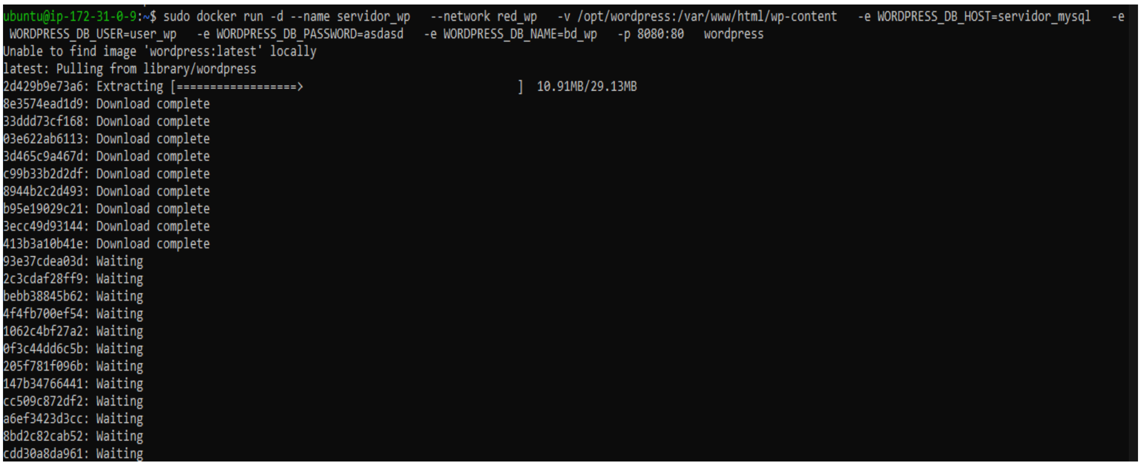
-e WORDPRESS_DB_USER=user_wp \

-e WORDPRESS_DB_PASSWORD=asdasd \

-e WORDPRESS_DB_NAME=bd_wp \

-p 8080:80 \

Wordpress

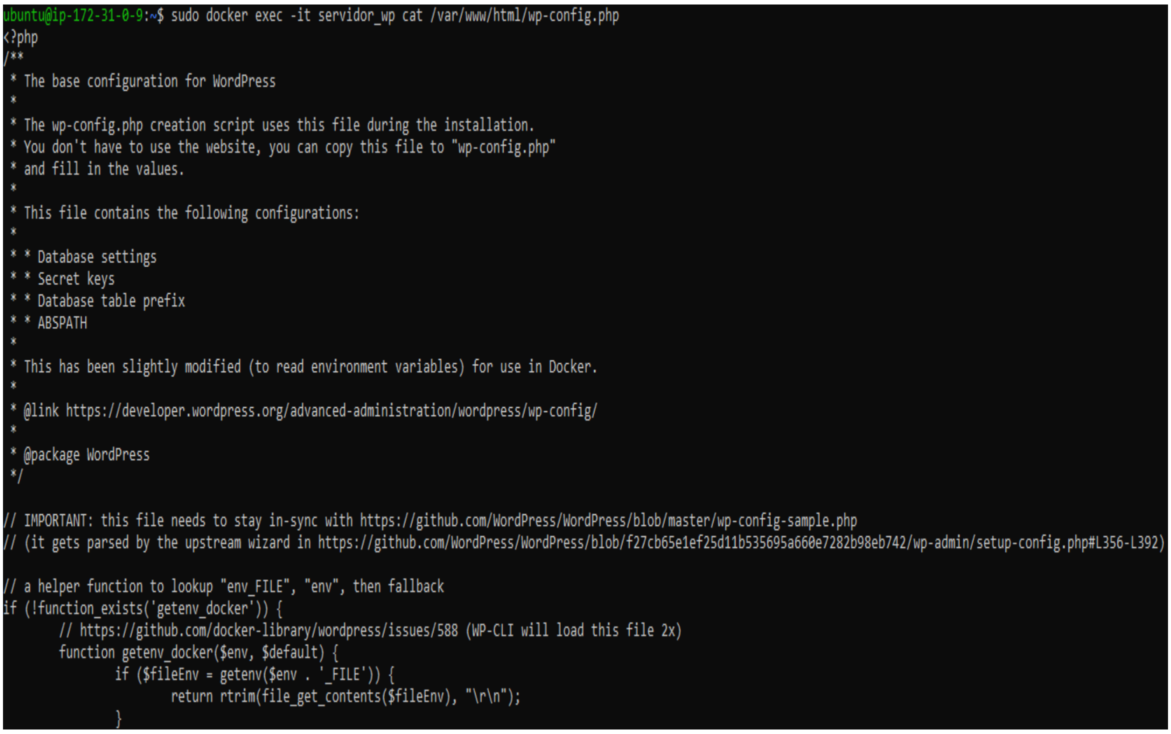


```
ubuntu@ip-172-31-0-9:~$ sudo docker run -d --name servidor_wp   --network red_wp   -v /opt/wordpress:/var/www/html/wp-content   -e WORDPRESS_DB_HOST=servidor_mysql   -e
 WORDPRESS_DB_USER=user_wp   -e WORDPRESS_DB_PASSWORD=asdasd   -e WORDPRESS_DB_NAME=bd_wp   -p 8080:80   wordpress
Unable to find image 'wordpress:latest' locally
latest: Pulling from library/wordpress
2d429b9e73a6: Extracting [==================>                                ]  10.91MB/29.13MB
8e3574ead1d9: Download complete
33ddd73cf168: Download complete
03e622ab6113: Download complete
3d465c9a467d: Download complete
c99b33b2d2df: Download complete
8944b2c2d493: Download complete
b95e19029c21: Download complete
3ecc49d93144: Download complete
413b3a10b41e: Download complete
93e37cdea03d: Waiting
2c3cdaf28ff9: Waiting
bebb38845b62: Waiting
4f4fb700ef54: Waiting
1062c4bf27a2: Waiting
0f3c44dd6c5b: Waiting
205f781f096b: Waiting
147b34766441: Waiting
cc509c872df2: Waiting
a6ef3423d3cc: Waiting
8bd2c82cab52: Waiting
cdd30a8da961: Waiting
```

7. Verificar el Archivo wp-config.php

Para visualizar el contenido del archivo de configuración, accede al contenedor servidor_wp y revisa el archivo:

docker exec -it servidor_wp cat /var/www/html/wp-config.php

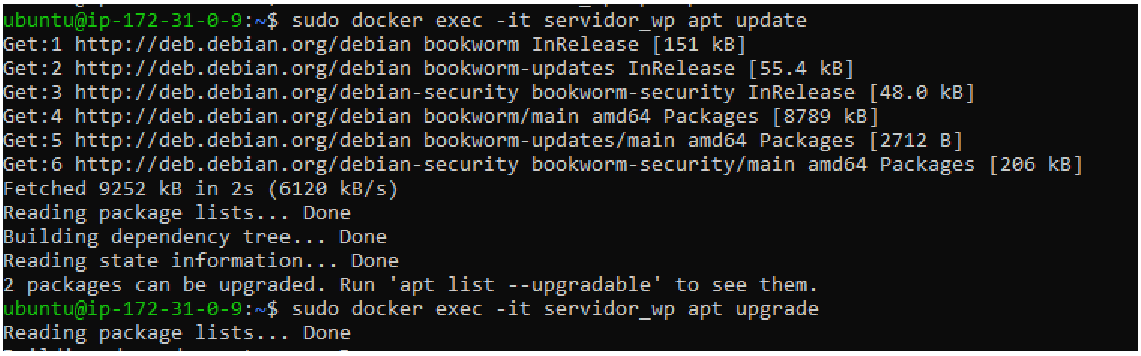


```
ubuntu@ip-172-31-0-9:~$ sudo docker exec -it servidor_wp cat /var/www/html/wp-config.php
<?php
/**
 * The base configuration for WordPress
 *
 * The wp-config.php creation script uses this file during the installation.
 * You don't have to use the website, you can copy this file to "wp-config.php"
 * and fill in the values.
 *
 * This file contains the following configurations:
 *
 * * Database settings
 * * Secret keys
 * * Database table prefix
 * * ABSPATH
 *
 * This has been slightly modified (to read environment variables) for use in Docker.
 *
 * @link https://developer.wordpress.org/advanced-administration/wordpress/wp-config/
 *
 * @package WordPress
 */

// IMPORTANT: this file needs to stay in-sync with https://github.com/WordPress/WordPress/blob/master/wp-config-sample.php
// (it gets parsed by the upstream wizard in https://github.com/WordPress/WordPress/blob/f27cb65e1ef25d11b535695a660e7282b98eb742/wp-admin/setup-config.php#L356-L392)

// a helper function to lookup "env_FILE", "env", then fallback
if (!function_exists('getenv_docker')) {
        // https://github.com/docker-library/wordpress/issues/588 (WP-CLI will load this file 2x)
        function getenv_docker($env, $default) {
                if ($fileEnv = getenv($env . '_FILE')) {
                        return rtrim(file_get_contents($fileEnv), "\r\n");
                }
```

8. Hacer Ping desde el Contenedor servidor_wp al servidor_mysql

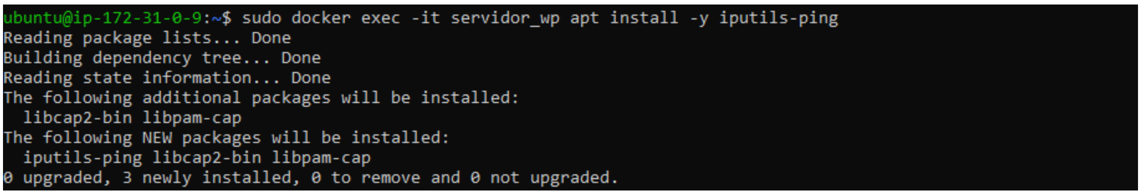
- Primero instala el paquete iputils-ping dentro del contenedor:

  docker exec -it servidor_wp apt update y luego el upgrade



```
ubuntu@ip-172-31-0-9:~$ sudo docker exec -it servidor_wp apt update
Get:1 http://deb.debian.org/debian bookworm InRelease [151 kB]
Get:2 http://deb.debian.org/debian bookworm-updates InRelease [55.4 kB]
Get:3 http://deb.debian.org/debian-security bookworm-security InRelease [48.0 kB]
Get:4 http://deb.debian.org/debian bookworm/main amd64 Packages [8789 kB]
Get:5 http://deb.debian.org/debian bookworm-updates/main amd64 Packages [2712 B]
Get:6 http://deb.debian.org/debian-security bookworm-security/main amd64 Packages [206 kB]
Fetched 9252 kB in 2s (6120 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
2 packages can be upgraded. Run 'apt list --upgradable' to see them.
ubuntu@ip-172-31-0-9:~$ sudo docker exec -it servidor_wp apt upgrade
Reading package lists... Done
```

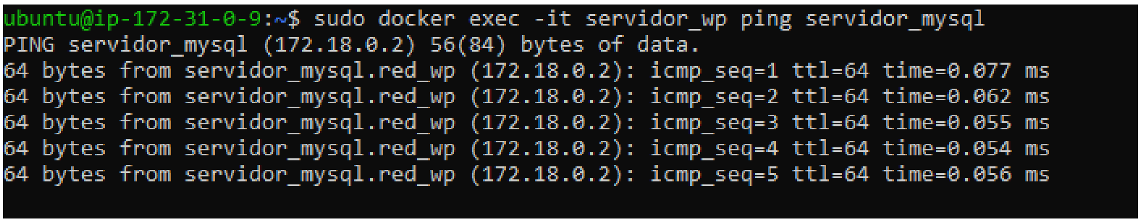
- **docker exec -it servidor_wp apt install -y iputils-ping**



```
ubuntu@ip-172-31-0-9:~$ sudo docker exec -it servidor_wp apt install -y iputils-ping
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libcap2-bin libpam-cap
The following NEW packages will be installed:
  iputils-ping libcap2-bin libpam-cap
0 upgraded, 3 newly installed, 0 to remove and 0 not upgraded.
```

- Luego, realiza un ping al contenedor servidor_mysql:

  docker exec -it servidor_wp ping servidor_mysql

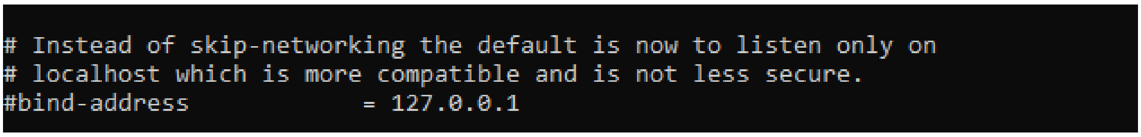


```
ubuntu@ip-172-31-0-9:~$ sudo docker exec -it servidor_wp ping servidor_mysql
PING servidor_mysql (172.18.0.2) 56(84) bytes of data.
64 bytes from servidor_mysql.red_wp (172.18.0.2): icmp_seq=1 ttl=64 time=0.077 ms
64 bytes from servidor_mysql.red_wp (172.18.0.2): icmp_seq=2 ttl=64 time=0.062 ms
64 bytes from servidor_mysql.red_wp (172.18.0.2): icmp_seq=3 ttl=64 time=0.055 ms
64 bytes from servidor_mysql.red_wp (172.18.0.2): icmp_seq=4 ttl=64 time=0.054 ms
64 bytes from servidor_mysql.red_wp (172.18.0.2): icmp_seq=5 ttl=64 time=0.056 ms
```

9. Revisar el Archivo de Configuración de MariaDB

Revisamos el parámetro bind-address en el archivo de configuración de MariaDB:

docker exec -it servidor_mysql cat /etc/mysql/mariadb.conf.d/50-server.cnf

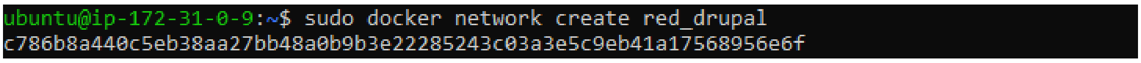


```
# Instead of skip-networking the default is now to listen only on
# localhost which is more compatible and is not less secure.
#bind-address            = 127.0.0.1
```

En este caso escucha a todas las interfaces

10. Desplegar Drupal

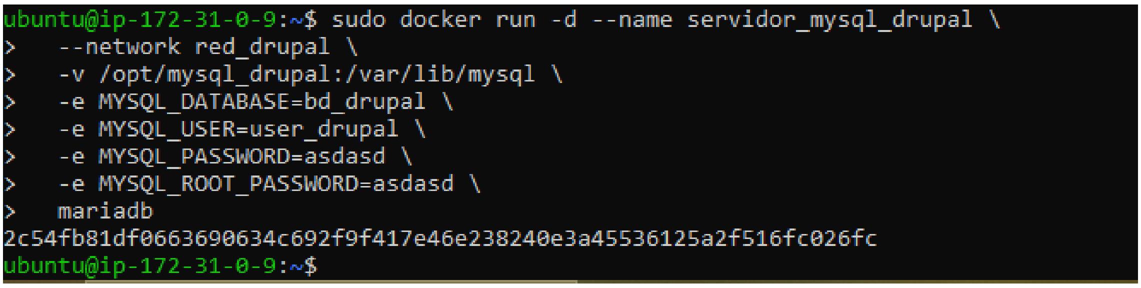
- Crea una red para Drupal:
  docker network create red_drupal



```
ubuntu@ip-172-31-0-9:~$ sudo docker network create red_drupal
c786b8a440c5eb38aa27bb48a0b9b3e22285243c03a3e5c9eb41a17568956e6f
```

- Despliega un contenedor MySQL para Drupal:

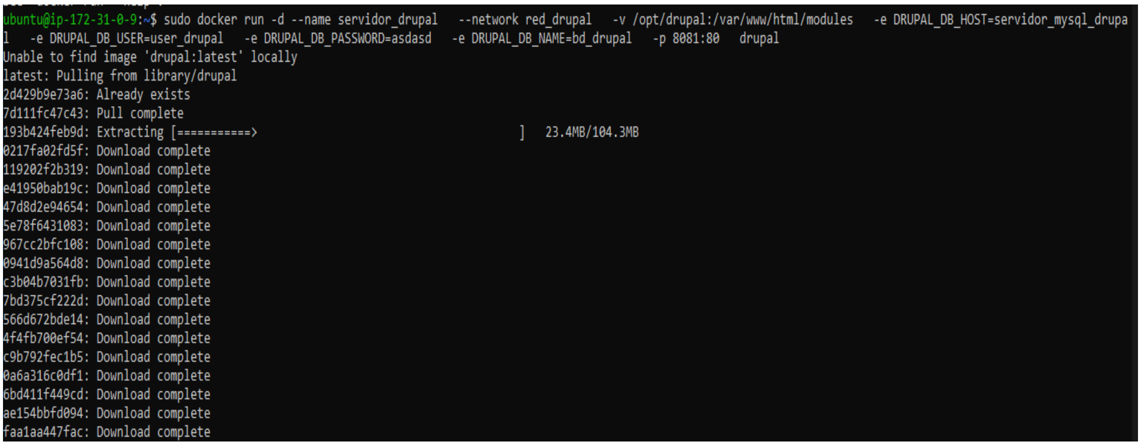
```
docker run -d --name servidor_mysql_drupal \
 --network red_drupal \
 -v /opt/mysql_drupal:/var/lib/mysql \
 -e MYSQL_DATABASE=bd_drupal \
 -e MYSQL_USER=user_drupal \
 -e MYSQL_PASSWORD=asdasd \
 -e MYSQL_ROOT_PASSWORD=asdasd \
 Mariadb
```



```
ubuntu@ip-172-31-0-9:~$ sudo docker run -d --name servidor_mysql_drupal \
>    --network red_drupal \
>    -v /opt/mysql_drupal:/var/lib/mysql \
>    -e MYSQL_DATABASE=bd_drupal \
>    -e MYSQL_USER=user_drupal \
>    -e MYSQL_PASSWORD=asdasd \
>    -e MYSQL_ROOT_PASSWORD=asdasd \
>    mariadb
2c54fb81df0663690634c692f9f417e46e238240e3a45536125a2f516fc026fc
ubuntu@ip-172-31-0-9:~$
```

- Despliega el contenedor Drupal:

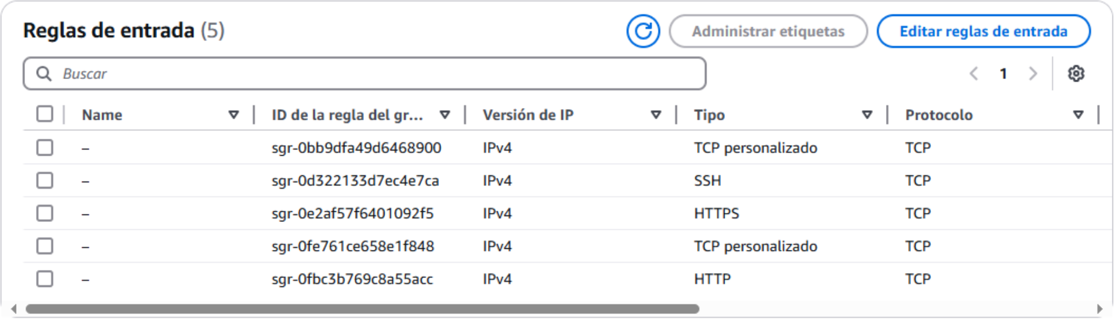
```
docker run -d --name servidor_drupal \
 --network red_drupal \
 -v /opt/drupal:/var/www/html/modules \
 -e DRUPAL_DB_HOST=servidor_mysql_drupal \
 -e DRUPAL_DB_USER=user_drupal \
 -e DRUPAL_DB_PASSWORD=asdasd \
 -e DRUPAL_DB_NAME=bd_drupal \
 -p 8081:80 \
 Drupal
```



```
ubuntu@ip-172-31-0-9:~$ sudo docker run -d --name servidor_drupal   --network red_drupal   -v /opt/drupal:/var/www/html/modules   -e DRUPAL_DB_HOST=servidor_mysql_drupa
l   -e DRUPAL_DB_USER=user_drupal   -e DRUPAL_DB_PASSWORD=asdasd   -e DRUPAL_DB_NAME=bd_drupal   -p 8081:80   drupal
Unable to find image 'drupal:latest' locally
latest: Pulling from library/drupal
2d429b9e73a6: Already exists
7d111fc47c43: Pull complete
193b424feb9d: Extracting [===========>                                       ]  23.4MB/104.3MB
0217fa02fd5f: Download complete
119202f2b319: Download complete
e41950bab19c: Download complete
47d8d2e94654: Download complete
5e78f6431083: Download complete
967cc2bfc108: Download complete
0941d9a564d8: Download complete
c3b04b7031fb: Download complete
7bd375cf222d: Download complete
566d672bde14: Download complete
4f4fb700ef54: Download complete
c9b792fec1b5: Download complete
0a6a316c0df1: Download complete
6bd411f449cd: Download complete
ae154bbfd094: Download complete
faa1aa447fac: Download complete
```

11. Verificar el Funcionamiento de WordPress y Drupal

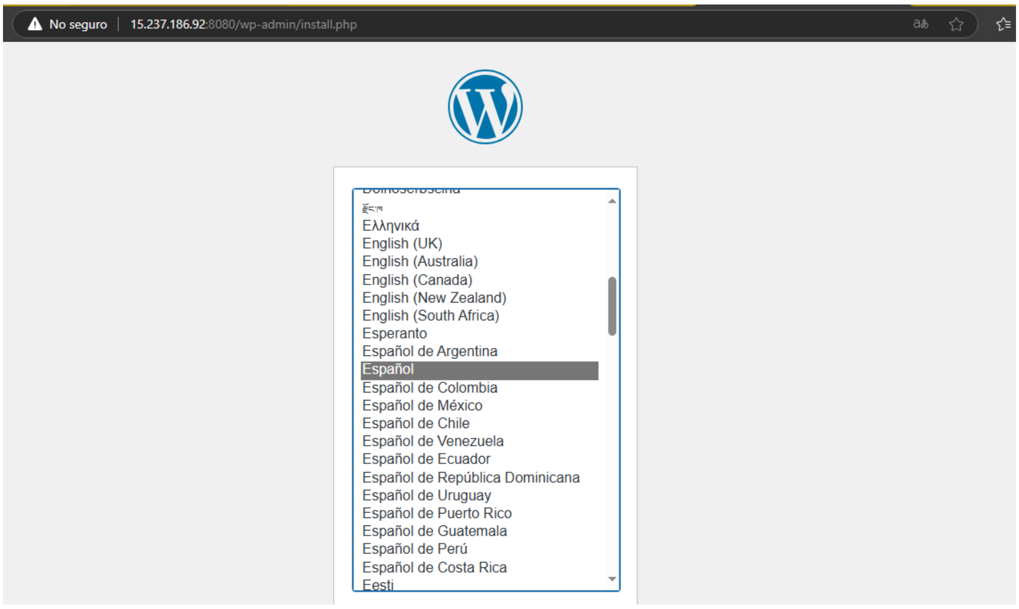
- Habilitar las reglas de entrada en el grupo de seguridad de AWS de nuestra instancia



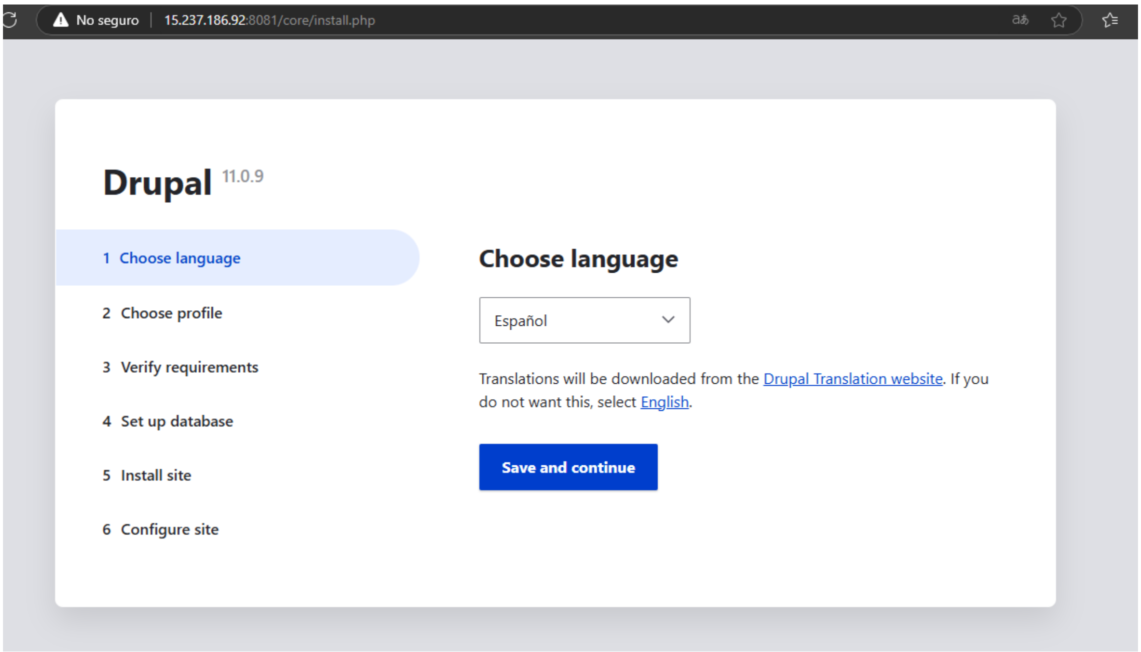
Reglas de entrada (5)

☐	Name	ID de la regla del gr...	Versión de IP	Tipo	Protocolo
☐	–	sgr-0bb9dfa49d6468900	IPv4	TCP personalizado	TCP
☐	–	sgr-0d322133d7ec4e7ca	IPv4	SSH	TCP
☐	–	sgr-0e2af57f6401092f5	IPv4	HTTPS	TCP
☐	–	sgr-0fe761ce658e1f848	IPv4	TCP personalizado	TCP
☐	–	sgr-0fbc3b769c8a55acc	IPv4	HTTP	TCP

- Verificar WordPress:
  Abre un navegador web e ingresa la dirección IP del host seguida del puerto 8080: http://15.237.186.92:8080.



- Verificar Drupal:
  Abre un navegador web e ingresa la dirección IP del host seguida del puerto 8081. Por ejemplo: http://15.237.186.92:8081.



Drupal 11.0.9

1 Choose language

2 Choose profile

3 Verify requirements

4 Set up database

5 Install site

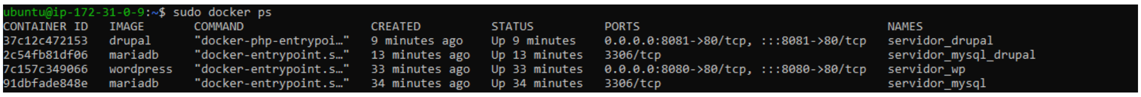
6 Configure site

Choose language

Español

Translations will be downloaded from the Drupal Translation website. If you do not want this, select English.

- Verificar Contenedores en Ejecución:
  Ejecuta el siguiente comando para verificar que todos los contenedores estén corriendo:
  docker ps



```
ubuntu@ip-172-31-0-9:~$ sudo docker ps
CONTAINER ID   IMAGE       COMMAND                  CREATED          STATUS          PORTS                                   NAMES
37c12c472153   drupal      "docker-php-entrypoi…"   9 minutes ago    Up 9 minutes    0.0.0.0:8081->80/tcp, :::8081->80/tcp   servidor_drupal
2c54fb81df06   mariadb     "docker-entrypoint.s…"   13 minutes ago   Up 13 minutes   3306/tcp                                servidor_mysql_drupal
7c157c349066   wordpress   "docker-entrypoint.s…"   33 minutes ago   Up 33 minutes   0.0.0.0:8080->80/tcp, :::8080->80/tcp   servidor_wp
91dbfade848e   mariadb     "docker-entrypoint.s…"   34 minutes ago   Up 34 minutes   3306/tcp                                servidor_mysql
```