# Guía Detallada de Despliegue de Aplicación PHP con Docker y OpenNebula

## 1. Introducción

Esta guía describe paso a paso el proceso para desplegar una aplicación PHP utilizando Docker y OpenNebula. El objetivo es proporcionar una solución de despliegue reproducible y fácil de mantener.

El proyecto esta alojado en: <http://172.20.228.140:8080/>

# Guía Detallada de Despliegue de Aplicación PHP con Docker y OpenNebula

## 1. Introducción

Esta guía describe paso a paso el proceso para desplegar una aplicación PHP utilizando Docker y OpenNebula. El objetivo es proporcionar una solución de despliegue reproducible y fácil de mantener.

## 2. Prerrequisitos

* **Cuenta de OpenNebula**: He usado un usuario que nos ha proporcionado Egibide para poder crear máquinas virtuales en OpenNebula.
* **Conocimientos**: Conocimientos básicos de Docker, PHP y la línea de comandos de Linux.
* **Software**:
  + Docker: Se ha utilizado la ultima versión de Docker. <https://docs.docker.com/engine/install/ubuntu/>
  + Docker Compose: Se ha utilizado para definir los servicios y networks.
* **Acceso a la Terminal**: Acceso a una terminal con SSH para conectarse a la máquina virtual en OpenNebula.

## 3. Configuración de la Máquina Virtual en OpenNebula

### Paso 1: Acceder a OpenNebula y Crear una Nueva Máquina Virtual

1. **Accede a OpenNebula**: Inicia sesión en tu cuenta de OpenNebula a través del panel web.
2. **Crear una Nueva VM**: Crear una maquina virtual Ubuntu o Debian. Dirígete a la sección "Plantillas" (En este caso por que ya tenemos plantillas proporcionadas) y selecciona "Ubuntu" o "Debian".
3. **Configurar los Recursos**: Configura los recursos de hardware para tu VM. Esto incluye el número de CPUs, la cantidad de RAM y el tamaño del disco. Para una aplicación pequeña, 2 CPUs y 2GB de RAM son suficientes pero en este caso he usado 2 CPUs físicas, 4 CPUs virtuales, 4GB de RAM y 25GB de almacenamiento.
4. **Configurar la Red**: Asigna una dirección IP estática a tu máquina virtual (opcional pero recomendado para un acceso consistente). Asegúrate de que la VM esté conectada a una red que tenga acceso a internet. He usado la tarjeta de red que se proporciona al usuario por defecto.
5. **Crear la VM**: Revisa la configuración y haz clic en "Create" para crear la máquina virtual.

### Paso 2: Conectar a la Máquina Virtual

1. Obtener la IP: Una vez que la VM esté en estado "Running", obtén su dirección IP desde el panel de control de OpenNebula.
2. Conectar por SSH: Utiliza SSH para conectarte a la máquina virtual desde tu terminal:

**ssh usuario@ip\_de\_la\_maquina\_virtual**

1. Reemplaza usuario con el nombre de usuario de la VM (por ejemplo, ubuntu) y ip\_de\_la\_maquina\_virtual con la dirección IP que obtuviste.  
   En mi caso he usado:

**ssh root@172.20.228.140**

## 4. Instalación de Docker en la Máquina Virtual

Para instalar Docker en la maquina virtual he seguido la guia que proporciona docker: <https://docs.docker.com/engine/install/ubuntu/>

1. Repositorio APT: Preparar repositorio APT:

**# Add Docker's official GPG key:**

**sudo apt-get update**

**sudo apt-get install 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**

**# Add the repository to Apt sources:**

**echo \ "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu \ $(. /etc/os-release && echo "${UBUNTU\_CODENAME:-$VERSION\_CODENAME}") stable" | \**

**sudo tee /etc/apt/sources.list.d/docker.list > /dev/null**

**sudo apt-get update**

1. Instalar Paquetes Docker: Instala Paquetes Docker utilizando el siguiente comando:

**sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin**

1. Verificar la Instalación: Verificar que Docker se haya instalado correctamente:

**docker --version**

o tambien se puede usar

sudo docker run hello-world

Deberías ver la versión de Docker instalada.

## 5. Configuración del Proyecto con Docker

Ahora, configuraremos tu proyecto PHP para que se pueda desplegar con Docker.

### Paso 1: Crear el Archivo Dockerfile

Crea un archivo llamado Dockerfile en la raíz de tu proyecto PHP con el siguiente contenido.

**FROM php:8.2-apache**

**# Actualiza e instala dependencias necesarias**

**RUN apt-get update && apt-get install -y \**

**$PHPIZE\_DEPS \**

**libzip-dev \**

**zip \**

**cron \**

**&& docker-php-ext-install pdo pdo\_mysql zip**

**# Establece el directorio de trabajo en el contenedor**

**WORKDIR /var/www/html**

**# Copia el código de la aplicación al contenedor**

**COPY . /var/www/html**

**# Crear la carpeta recursos/images y asignar permisos**

**RUN mkdir -p /var/www/html/recursos/images && \**

**chown -R www-data:www-data /var/www/html && \**

**chmod -R 775 /var/www/html**

**# Opcional: configura ServerName para eliminar la advertencia de Apache**

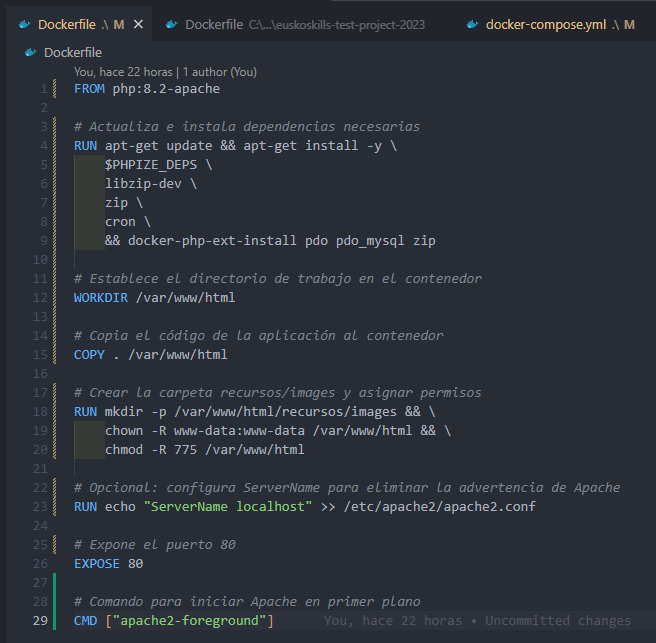
**RUN echo "ServerName localhost" >> /etc/apache2/apache2.conf**

**# Expone el puerto 80**

**EXPOSE 80**

**# Comando para iniciar Apache en primer plano**

**CMD ["apache2-foreground"]**

****

### Paso 2: Crear el Archivo docker-compose-yml

Crea un archivo llamado Dockerfile en la raíz de tu proyecto PHP con el siguiente contenido. **services:**

**web:**

**build: .**

**ports:**

**- "8080:80"**

**volumes:**

**- .:/var/www/html**

**environment:**

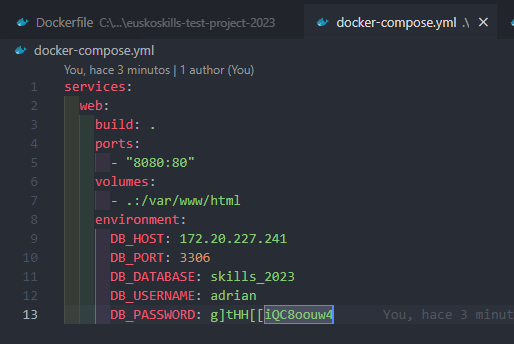
**DB\_HOST: 172.20.227.241**

**DB\_PORT: 3306**

**DB\_DATABASE: skills\_2023**

**DB\_USERNAME: adrian**

**DB\_PASSWORD: g]tHH[[iQC8oouw4**

****

### Paso 3: Ajustar Index.php

He usado varios ucfirst() y un lcfirst() para ajustar las mayusculas y minusculas ya que Linux es case sensitive.

**<?php**

**require\_once 'config/config.php';**

**require\_once 'model/db.php';**

**if(!isset($\_GET["controller"])) $\_GET["controller"] = constant("DEFAULT\_CONTROLLER");**

**if(!isset($\_GET["action"])) $\_GET["action"] = constant("DEFAULT\_ACTION");**

**$controller\_path = 'controller/' . ucfirst($\_GET["controller"]) . 'Controller.php';**

**if(!file\_exists($controller\_path)) $controller\_path = 'controller/' . constant("DEFAULT\_CONTROLLER") . '.php';**

**require\_once $controller\_path;**

**$controllerName = ucfirst($\_GET["controller"]) . 'Controller';**

**$controller = new $controllerName();**

**$dataToView["data"] = array();**

**if(method\_exists($controller, $\_GET["action"])) $dataToView["data"] = $controller -> {$\_GET["action"]}();**

**require\_once 'view/layout/header.php';**

**require\_once 'view/' . lcfirst($\_GET["controller"]) . '/' . $controller -> view . '.html.php';**

**require\_once 'view/layout/footer.php';**

**?>**

## 6. Despliegue de la Aplicación

Una vez que hayas configurado tu máquina virtual OpenNebula con Docker y preparado tu proyecto con los archivos Dockerfile y docker-compose.yml, estarás listo para desplegar tu aplicación. Esta sección te guía a través de ese proceso.

Requisitos previos:

* Asegúrate de haber completado los pasos de las secciones 3, 4 y 5.
* Verifica que tienes acceso a la terminal de la máquina virtual a través de SSH.
* Comprueba que Docker y Docker Compose están instalados correctamente en la máquina virtual.

### Paso 1: Clonar el Repositorio del Proyecto

1. Conéctate a la VM (si aún no lo estás): Utiliza SSH para conectarte a tu máquina virtual OpenNebula.

**ssh usuario@ip\_de\_la\_maquina\_virtual**

**ssh root@172.20.228.140**

1. Reemplaza usuario con el nombre de usuario (por ejemplo, ubuntu) y ip\_de\_la\_maquina\_virtual con la dirección IP de tu VM.
2. Clona el Repositorio: Clona el repositorio de tu proyecto PHP desde GitHub utilizando el comando git clone:

**git clone https://github.com/egibide-daw/euskoskills-testproject2023-adriloma21**

1. Navega al Directorio del Proyecto: Una vez que el repositorio se haya clonado, navega al directorio del proyecto:

**cd euskoskills-testproject2023-adriloma21**

### Paso 2: Construir y Ejecutar los Contenedores de Docker

1. Construir las Imágenes: Conéctate a la VM (si aún no lo estás) y navega al directorio del proyecto que clonaste. Construye las imágenes de Docker utilizando Docker Compose. Este comando leerá el archivo docker-compose.yml y seguirá las instrucciones del Dockerfile para construir la imagen de tu aplicación PHP.

**docker compose build**

1. Ejecutar los Contenedores: Una vez que las imágenes se hayan construido correctamente, ejecuta los contenedores en modo detached (en segundo plano) usando Docker Compose:

**docker compose up -d**

1. Este comando iniciará los contenedores definidos en el docker-compose.yml. El parámetro -d indica que los contenedores se ejecutarán en segundo plano, liberando la terminal para otros comandos.
   * **Información:** Docker Compose creará una red virtual y conectará los contenedores entre sí. También mapeará el puerto 80 del contenedor al puerto 8080 de la máquina virtual, lo que permitirá acceder a la aplicación desde un navegador web.

### Paso 3: Verificar que la Aplicación se Está Ejecutando

1. Comprobar el Estado: Después de ejecutar el comando **docker-compose up -d**, verifica que los contenedores se estén ejecutando correctamente usando el comando docker-compose ps:

**docker compose ps**

1. Deberías ver una lista de los contenedores en ejecución con su estado. Asegúrate de que el estado de todos los contenedores sea "Up".
   * Salida de Ejemplo:

CONTAINER ID IMAGE COMMAND CREATED ---------------------------------------------------------------------------------------------------------------------------e0228aa87719 euskoskills-test-project-2023-web "docker-php-entrypoi…" 2 hours ago

STATUS PORTS -------------------------------------------------------------------------------------------------------------------------

Up 2 hours 0.0.0.0:8080->80/tcp, [::]:8080->80/tcp

NAMES

—-------------------------------------------

euskoskills-test-project-2023-web-1

1. Acceder a la Aplicación: Abre tu navegador web e introduce la dirección IP de tu máquina virtual seguida del puerto 8080 (por ejemplo, **http://172.20.228.140:8080**).  
   Si todo está configurado correctamente, deberías ver la página de inicio de tu aplicación PHP.
2. Verificar los Logs (en caso de problemas): Si tienes problemas para acceder a la aplicación, revisa los logs del contenedor web para identificar posibles errores:

**docker-compose logs web**

1. Este comando mostrará los logs del contenedor web, que pueden contener información útil para solucionar problemas. Busca errores PHP, errores de conexión a la base de datos, etc.