



Manual de Despliegue en Docker

para Sistema de Alquiler de Barcos

Este documento presenta las instrucciones necesarias para el despliegue del sistema de alquiler de barcos, detallando los pasos para su instalación, configuración y operación, con el objetivo de garantizar su correcto funcionamiento.

Equipo de Desarrollo - Grupo G3.2

- Franco Dell Aguila Ureña
- Enrique Anda Hernández
- Yesica Garate Fuentes
- Pablo Montero Rollán
- Pedro Oliva Rodríguez

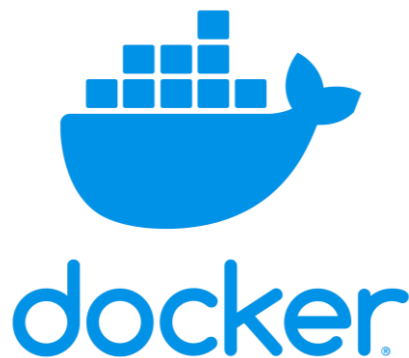
CONTENIDO

INTRODUCCIÓN	3
¿Qué es Docker?.....	3
Propósito del despliegue con Docker para este proyecto	3
REQUISITOS PREVIOS	4
Software necesario:	4
Archivos requeridos en el proyecto:.....	4
ESTRUCTURA DEL PROYECTO	5
Organización de los directorios del proyecto.....	5
PASOS PARA EL DESPLIEGUE	6
Configuración del archivo Dockerfile	6
Configuración del archivo docker-compose.yml.....	7
Modificación en la configuración de nuestro settings.py	7
Construcción de la imagen de Docker.	7
Ejecución del contenedor.....	9
SOLUCIÓN DE PROBLEMAS	10
Errores comunes y cómo resolverlos.	10
REFERENCIAS	11

INTRODUCCIÓN

¿Qué es Docker?

Docker es una plataforma de software que permite crear, implementar y ejecutar aplicaciones dentro de contenedores. Un contenedor es una unidad ligera y autónoma que empaqueta una aplicación junto con todas sus dependencias (librerías, herramientas del sistema, código, configuraciones), lo que asegura que funcione de manera consistente en cualquier entorno.



Propósito del despliegue con Docker para este proyecto

Uno de los beneficios que nos ofrece Docker es que puede ejecutarse en cualquier entorno que tenga Docker instalado, sin importar el sistema operativo o la configuración del servidor o los conflictos de versiones o actualizaciones inesperadas.

El despliegue es rápido y confiable, lo que permitirá que los usuarios finales disfruten de una experiencia sin interrupciones además de que en caso de alta escalabilidad (si tenemos más usuarios reservando barcos) podríamos ir añadiendo contenedores adicionales.

REQUISITOS PREVIOS

Software necesario:

1. Docker

- Versión recomendada: Docker Desktop (para Windows o MacOS) o Docker Engine (para Linux). Descarga desde: <https://www.docker.com/get-started>

2. Docker compose (incluido con Docker Desktop en versiones recientes)

- Usado para orquestar múltiples contenedores, como tu aplicación y la base de datos.

3. Editor de texto o IDE (opcional, para modificar configuraciones si es necesario)

- Recomendado: Visual Studio Code o cualquier editor que soporte YAML y Dockerfiles.

4. Git (opcional, si descargas el proyecto desde un repositorio)

- Usado para clonar el proyecto desde un repositorio como GitHub

Archivos requeridos en el proyecto:

Estos son los archivos imprescindibles para asegurar un despliegue exitoso con Docker. Verifica que estén presentes y correctamente configurados en el proyecto; de lo contrario, deberás crearlos y/o ajustarlos según los requisitos de la aplicación.

1. Dockerfile: Archivo que define cómo se construye la imagen Docker de la aplicación.

2. docker-compose.yml: Archivo que define cómo interactúan los contenedores, como la aplicación y la base de datos.

3. Código Fuente de la Aplicación: Todos los archivos necesarios para la aplicación, incluyendo:

- Código python
- Configuraciones en settings.py
- Dependencias: archivo requirements.txt

4. Archivos Estáticos: Recursos como imágenes, CSS o JavaScript, asegúrate de incluirlos.

5. Archivo de Configuración de Base de Datos: Credenciales o configuraciones necesarias para conectar la aplicación a la base de datos.

ESTRUCTURA DEL PROYECTO

Organización de los directorios del proyecto.

El proyecto está organizado de la siguiente manera:

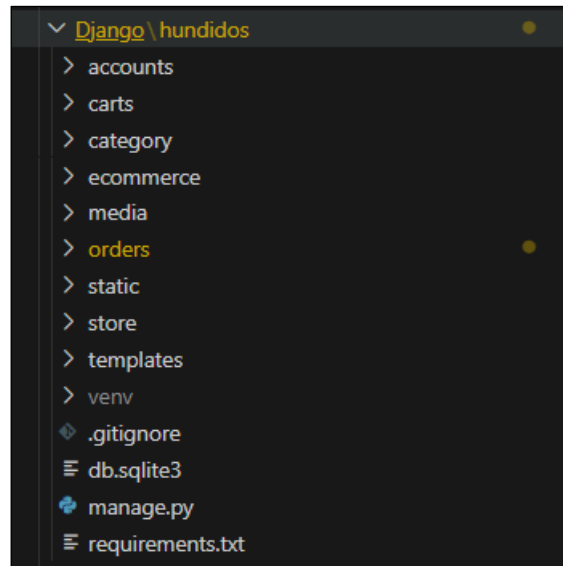


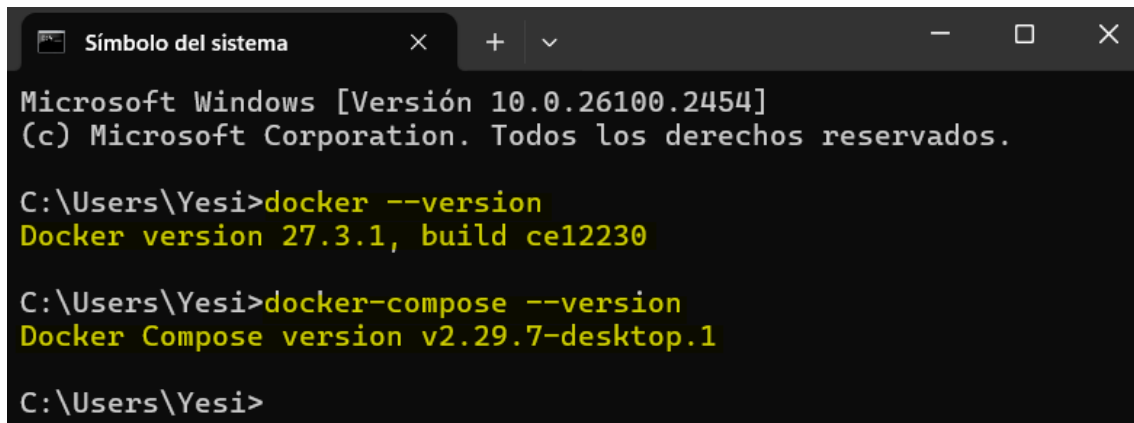
Ilustración 1. Estructura de Sistema de Alquiler de barcos Hundidos S.L.

1. **requirements.txt:** Contiene las dependencias necesarias para el proyecto. Aquí se podrá actualizar las bibliotecas requeridas, como Django y otras dependencias en el futuro.
2. **media/:** En esta carpeta se almacenan los archivos tales como imágenes de productos, banners, icons, etc.
3. **Código Fuente:** Las aplicaciones están correctamente organizadas en directorios como accounts/, carts/, orders/, templates/, etc. Contienen el código principal de la aplicación, no es necesario mover ni modificar la estructura.
4. **db.sqlite3:** Contiene la base de datos SQLite del proyecto (ideal para desarrollo).
5. **manage.py:** Es el punto de entrada principal para ejecutar comandos de Django.
6. **.gitignore:** Esta configurado para ignorar ciertos archivos o carpetas.

Los archivos como Dockerfile y docker-compose.yml deben ser creados en la raíz del proyecto junto a archivos como manage.py, gitignore, etc.

PASOS PARA EL DESPLIEGUE

Para verificar si Docker y docker-compose están correctamente instalados en tu sistema, abre una terminal y ejecuta los siguientes comandos:



```
Microsoft Windows [Versión 10.0.26100.2454]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\Yesi>docker --version
Docker version 27.3.1, build ce12230

C:\Users\Yesi>docker-compose --version
Docker Compose version v2.29.7-desktop.1

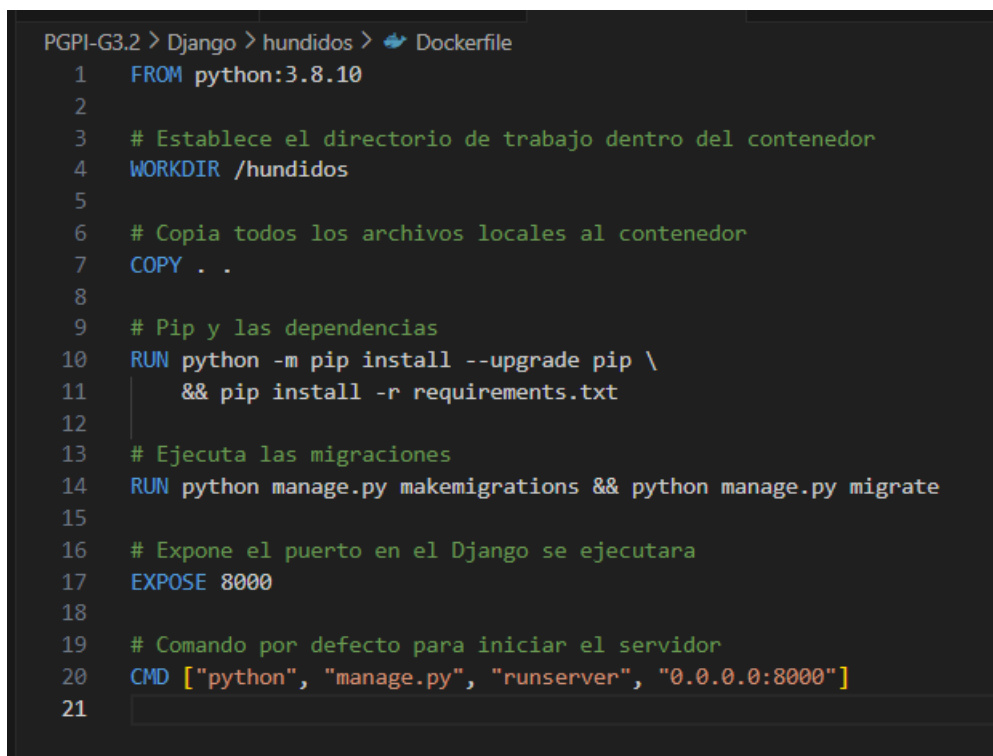
C:\Users\Yesi>
```

Ilustración 2. Comandos para verificación de instalación de docker y docker-compose

Verificar la instalación del editor de texto de su preferencia.

Configuración del archivo Dockerfile

Si no esta presente lo creamos en la raíz del proyecto, un archivo sin extensión con la siguiente estructura.



```
PGPI-G3.2 > Django > hundidos > Dockerfile
1 FROM python:3.8.10
2
3 # Establece el directorio de trabajo dentro del contenedor
4 WORKDIR /hundidos
5
6 # Copia todos los archivos locales al contenedor
7 COPY . .
8
9 # Pip y las dependencias
10 RUN python -m pip install --upgrade pip \
11     && pip install -r requirements.txt
12
13 # Ejecuta las migraciones
14 RUN python manage.py makemigrations && python manage.py migrate
15
16 # Expone el puerto en el Django se ejecutara
17 EXPOSE 8000
18
19 # Comando por defecto para iniciar el servidor
20 CMD ["python", "manage.py", "runserver", "0.0.0.0:8000"]
21
```

Ilustración 3. Estructura del archivo Dockerfile

RECUERDA que no existe un único estilo de Dockerfile para cada proyecto, pero sí hay patrones comunes. La estructura y contenido de un Dockerfile pueden variar para optimizar rendimiento, tamaño, o para incluir configuraciones específicas.

Configuración del archivo docker-compose.yml

En la raíz del proyecto, junto al archivo Dockerfile, se debe crear el archivo docker-compose.yml. Este archivo debe tener la siguiente estructura:

```
PGPI-G3.2 > Django > hundidos > 📄 docker-compose.yml
1  version: "3.9"
2
3  services:
4    web:
5      build:
6        context: .
7        dockerfile: Dockerfile
8      volumes:
9        - ../hundidos
10     ports:
11       - "8000:8000"
12     environment:
13       - DEBUG=1
14       - DJANGO_SETTINGS_MODULE=ecommerce.settings
15     command: ["python", "manage.py", "runserver", "0.0.0.0:8000"]
16
17   volumes:
18     sqlite_data:
```

Ilustración 4. Estructura del archivo dockerfile-compose

Modificación en la configuración de nuestro settings.py

Después de completar la configuración anterior, es necesario realizar una pequeña modificación en el archivo settings.py. Debemos asegurarnos de que localhost esté incluido en la lista correspondiente (como ALLOWED_HOSTS). Si ya está incluido, no es necesario realizar cambios; de lo contrario, lo agregamos.

```
37
38  ALLOWED_HOSTS = ['alquiler-de-barcos.onrender.com', 'localhost', 'pgpi-g3-2.onrender.com', '127.0.0.1']
39
```

Ilustración 5. Incluir localhost en ALLOWED_HOSTS de settings.py

Construcción de la imagen de Docker.

Después de haber creado los archivos necesarios, puedes proceder a construir la imagen de Docker desde la carpeta raíz del proyecto. Para ello, utiliza el siguiente comando:

docker-compose build

Este comando utiliza docker-compose para leer el archivo docker-compose.yml y construir las imágenes definidas en él. Durante el proceso, se leerá el Dockerfile asociado para crear la imagen especificada, así como cualquier otra imagen adicional que sea requerida.

```
PS C:\Users\Yesi\Desktop\barcos\PGPI-G3.2\Django\hundidos> docker-compose build
time="2024-12-01T19:37:33+01:00" level=warning msg="C:\Users\Yesi\Desktop\barcos\PGPI-G3.2\Django\h
undidos\docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to
avoid potential confusion"
[+] Building 16.2s (9/10)                                docker:desktop-linux
=> [web internal] load metadata for docker.io/library/python:3.8.10      1.6s
=> [web auth] library/python:pull token for registry-1.docker.io         0.0s
=> [web internal] load .dockerignore                                     0.1s
=> => transferring context: 2B                                           0.0s
=> [web 1/5] FROM docker.io/library/python:3.8.10@sha256:f7981c32c931f07d053f6867d7882169a97695286 0.0s
=> [web internal] load build context                                    1.1s
=> => transferring context: 1.43MB                                       1.0s
=> CACHED [web 2/5] RUN apt-get update && apt-get install -y chromium chromium-driver 0.0s
=> CACHED [web 3/5] WORKDIR /hundidos                                   0.0s
=> [web 4/5] COPY . .                                                  3.4s
=> [web 5/5] RUN python -m pip install --upgrade pip && pip install -r requirements.txt 10.0s
=> => # Collecting websocket-client~=1.8 (from selenium->-r requirements.txt (line 17))
=> => #   Downloading websocket_client-1.8.0-py3-none-any.whl.metadata (8.0 kB)
```

Ilustración 6. Ejecución del comando para crear una imagen en docker

Esto incluye la instalación de dependencias, copiado de archivos y configuraciones necesarias para que el contenedor funcione correctamente. Asegúrate de que no aparezcan errores durante este proceso.

Para confirmar que la imagen se ha creado correctamente, utilizamos el siguiente comando:

docker images

```
PS C:\Users\Yesi\Desktop\barcos\PGPI-G3.2\Django\hundidos> docker images
REPOSITORY          TAG             IMAGE ID        CREATED         SIZE
hundidos-web        latest          96c4a877d456   About a minute ago  1.69GB
<none>              <none>          a5c98362ea32   6 hours ago     1.49GB
PS C:\Users\Yesi\Desktop\barcos\PGPI-G3.2\Django\hundidos>
```

Ilustración 7. Comandos en docker para listar las imágenes creadas en docker

Este comando mostrará una lista de todas las imágenes disponibles en tu entorno de Docker. En esta lista, deberías encontrar una imagen con el nombre **hundidos-web**, confirmando que se creó correctamente.

Ejecución del contenedor

Si todo se ha configurado correctamente, podemos iniciar los servicios definidos en el archivo docker-compose.yml utilizando el siguiente comando:

docker-compose up

Este comando levantará los contenedores definidos en el archivo docker-compose.yml, incluido el contenedor que creaste anteriormente. Si deseas ejecutar los servicios en segundo plano (modo "detached"), puedes agregar la opción -d al comando:

```
PS C:\Users\Yesi\Desktop\barcos\PGPI-G3.2\Django\hundidos> docker-compose up
time="2024-12-01T19:43:43+01:00" level=warning msg="C:\\Users\\Yesi\\Desktop\\barcos\\PGPI-G3.2\\Django\\
\\hundidos\\docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it
to avoid potential confusion"
[+] Running 2/2
 ✓ Network hundidos_default Created                                0.1s
 ✓ Container hundidos-web-1 Created                                0.2s
Attaching to web-1
█
```

Ilustración 8. Comando para levantar servicios en docker

Una vez en ejecución, podrás acceder a la aplicación desde tu navegador ingresando a <http://localhost:8000>. Asegúrate de que el puerto 8000 esté disponible en tu máquina para evitar conflictos.

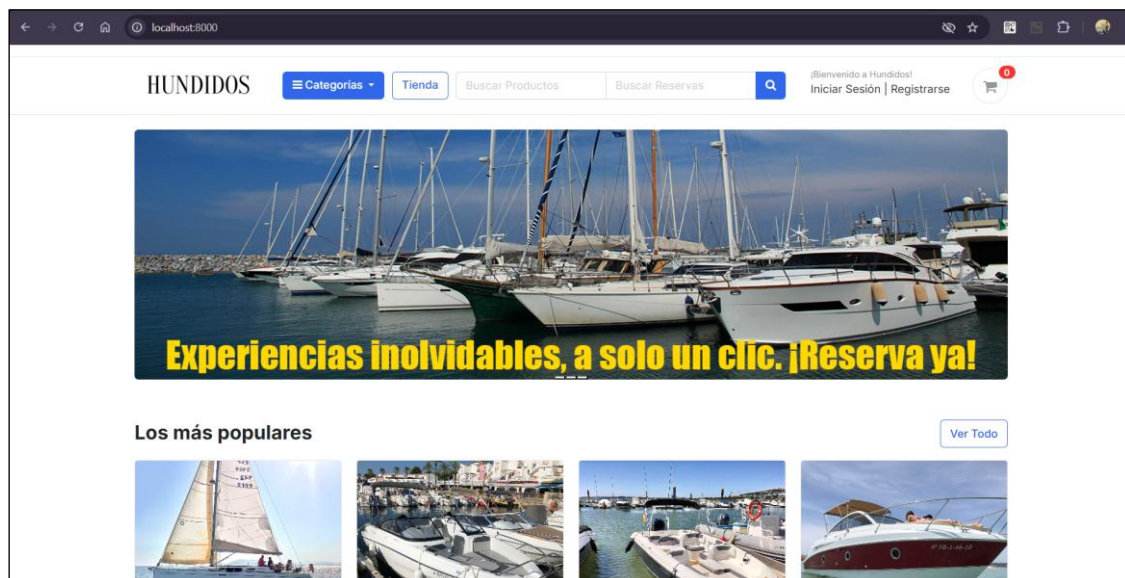


Ilustración 9. Aplicación levanta usando contenedores de docker

SOLUCIÓN DE PROBLEMAS

Errores comunes y cómo resolverlos.

Al ingresar al navegador a <http://localhost:8000>

DisallowedHost at /

Invalid HTTP_HOST header: 'localhost:8000'. You may need to add 'localhost' to ALLOWED_HOSTS.

```
Request Method: GET
Request URL: http://localhost:8000/
Django Version: 3.2.20
Exception Type: DisallowedHost
Exception Value: Invalid HTTP_HOST header: 'localhost:8000'. You may need to add 'localhost' to ALLOWED_HOSTS.
Exception Location: /usr/local/lib/python3.8/site-packages/django/http/request.py, line 151, in get_host
Python Executable: /usr/local/bin/python
Python Version: 3.8.10
Python Path: ['/hundreds',
              '/usr/local/lib/python3.8.zip',
              '/usr/local/lib/python3.8',
              '/usr/local/lib/python3.8/lib-dynload',
              '/usr/local/lib/python3.8/site-packages']
Server time: Fri, 29 Nov 2024 10:52:47 +0000
```

Este error indica que la configuración de Django no permite el acceso desde localhost:8000. Esto se debe a que no se ha agregado en la variable ALLOWED_HOSTS del archivo de configuración (settings.py) localhost. Para solucionarlo, sigue estos pasos:

1. Editar ALLOWED_HOSTS en settings.py: Abre el archivo settings.py de tu proyecto Django (normalmente está en la carpeta principal del proyecto). Busca la línea que define ALLOWED_HOSTS y cámbiala para incluir localhost. Debería quedar algo así:

```
ALLOWED_HOSTS = ['localhost', '127.0.0.1', '0.0.0.0']
```

REFERENCIAS

1. *Manuals.* (s.f.-b). Docker Documentation. <https://docs.docker.com/manuals/>