

Universidad de San Carlos de Guatemala  
Facultad de Ingeniería  
Escuela de Mecánica Eléctrica  
Proyectos Aplicados A I.E  
Ing. José Anibal Silva de los Angeles  
Sección P



Tarea No.9

Fecha de entrega 20/10/2024

Integrante	Registro académico
Emerson Aldair Pérez Rivera	201902852
Johann José Miguel Escobar González	202100140

---

## Tarea 09: Docker

En este reporte, se analizará el uso de Docker para dockerizar un proyecto desarrollado en Django, un popular framework de desarrollo web en Python. Docker ofrece una solución eficaz para crear entornos de desarrollo y producción consistentes mediante contenedores ligeros y portátiles. A través de la contenedorización, se facilita la gestión de dependencias, se mejora la escalabilidad y se simplifica la configuración del entorno. Esto no solo optimiza el flujo de trabajo del desarrollo, sino que también garantiza que la aplicación funcione de manera idéntica en diferentes entornos.

### I. MARCO TEORICO

#### A. Docker

Docker utiliza contenedores para empaquetar una aplicación y sus dependencias en un solo paquete portátil. Los contenedores son ligeros, rápidos y pueden ser creados, destruidos y replicados fácilmente.

#### B. Imagen de Docker

Una imagen de Docker es una plantilla inmutable que contiene todo lo necesario para ejecutar una aplicación: código, dependencias, bibliotecas, etc. Las imágenes son el punto de partida para crear contenedores.

#### C. Contenedor de Docker

Un contenedor es una instancia en ejecución de una imagen de Docker. Cada contenedor es una entidad aislada que se ejecuta de manera independiente, pero puede comunicarse con otros contenedores a través de redes definidas.

#### D. Ventajas de Docker

Docker ofrece varias ventajas, entre las que se incluyen:

- **Portabilidad:** Las aplicaciones dockerizadas pueden ejecutarse en cualquier sistema que tenga Docker instalado.
- **Consistencia:** Los contenedores aseguran que el entorno de desarrollo, pruebas y producción sea el mismo.
- **Escalabilidad:** Docker permite escalar aplicaciones de manera fácil y eficiente.
- **Aislamiento:** Cada contenedor es independiente y aislado, lo que mejora la seguridad y la estabilidad.

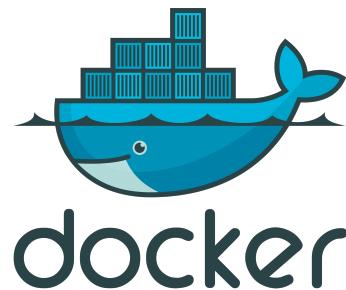


Figura 1

Fuente: Elaboracion Propia, 2024

### II. INSTALAR USANDO EL REPOSITORIO APT

Antes de instalar Docker Engine por primera vez en una nueva máquina host, debe configurar el repositorio apt de Docker. Luego, puede instalar y actualizar Docker desde el repositorio.

#### A. Configurar el repositorio apt de Docker

##### Agregar la clave GPG oficial de Docker

```

1 sudo apt-get update
2 sudo apt-get install ca-certificates curl
3 sudo install -m 0755 -d /etc/apt/keyrings
4 sudo curl -fsSL https://download.docker.com/
   linux/debian/gpg -o /etc/apt/keyrings/docker
   .asc
5 sudo chmod a+r /etc/apt/keyrings/docker.asc

```

#### B. Agregar el repositorio a las fuentes de apt

```

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

```

**Nota:** Si usa una distribución derivada, como Kali Linux, es posible que deba sustituir la parte de este comando que se espera que imprima el nombre del código de la versión:

```
1 (. /etc/os-release && echo "$VERSION_CODENAME")
```

Reemplace esta parte con el nombre en código de la versión de Debian correspondiente, como *bookworm*.

### C. Instalar los paquetes de Docker

#### Versión más reciente

Para instalar la última versión, ejecute:

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

### D. Verificar la instalación

Para verificar que la instalación sea exitosa, ejecute la imagen *hello-world*:

```
1 sudo docker run hello-world
```

Este comando descarga una imagen de prueba y la ejecuta en un contenedor. Cuando el contenedor se ejecuta, imprime un mensaje de confirmación y sale.

## III. CODIGOS DOCKER

### A. requirements.txt

```
1 assigref==3.8.1
2 Brotli==1.1.0
3 certifi==2024.8.30
4 cffi==1.17.0
5 charset-normalizer==3.4.0
6 click==8.1.7
7 cmake==3.30.5
8 colorama==0.4.6
9 contourpy==1.2.1
10 cssselect2==0.7.0
11 cycler==0.12.1
12 Django==5.1.1
13 django-adminlte3==0.1.6
14 django-axes==6.5.2
15 django-grappelli==4.0.1
16 django-recaptcha==4.0.0
17 django-suit==0.2.28
18 django-widget-tweaks==1.5.0
19 dlib==19.24.6
20 face-recognition==1.3.0
21 face-recognition-models @ git+https://github.com
   /ageitgey/face_recognition_models.
   git@e67de717267507d1e9246de95692eb8be736ab61
22 fonttools==4.53.1
23 gobject==0.1.0
24 html5lib==1.1
25 idna==3.10
26 imutils==0.5.4
27 kiwisolver==1.4.5
28 matplotlib==3.9.1
29 numpy==2.0.1
30 opencv-contrib-python==4.10.0.84
```

```
31 packaging==24.1
32 pdfkit==1.0.0
33 pillow==10.4.0
34 psycopg2==2.9.9
35 pycparser==2.22
36 pydyf==0.11.0
37 pyparsing==3.1.2
38 pyphen==0.16.0
39 python-dateutil==2.9.0.post0
40 python-decouple==3.8
41 requests==2.32.3
42 scipy==1.14.0
43 six==1.16.0
44 sounddevice==0.5.0
45 sqlparse==0.5.1
46 tinycc==1.3.0
47 tzdata==2024.1
48 urllib3==2.2.3
49 Wave==0.0.2
50 weasyprint==62.3
51 webencodings==0.5.1
52 zopfli==0.2.3
```

### B. Dockerfile

```
1 FROM python:3.10-slim
2 ENV PYTHONUNBUFFERED 1
3 RUN apt-get update && apt-get install -y libpq-
  dev gcc cmake weasyprint libglib-mesa-glx
  libglib2.0-0 git
4 WORKDIR /app
5 COPY requirements.txt .
6 RUN python -m pip install --upgrade pip
7 RUN python -m pip install -r requirements.txt
8 COPY . .
9 EXPOSE 8000
10 CMD ["python", "manage.py", "runserver",
      "0.0.0.0:8000"]
```

### C. Docker Compose

```
1 version: '3'
2 services:
3   db:
4     image: postgres:13
5     environment:
6       POSTGRES_DB: parcial2
7       POSTGRES_USER: emerson
8       POSTGRES_PASSWORD: kuto
9     volumes:
10       - postgres_data:/var/lib/postgresql/data
11     networks:
12       - mynetwork
13   web:
14     build: .
15     command: python manage.py runserver
16     0.0.0.0:8000
17     volumes:
18       - ./app
19     ports:
20       - "8000:8000"
21     depends_on:
22       - db
23     networks:
```

```

23     - mynetwork
24 networks:
25   mynetwork:
26 volumes:
27   postgres_data:

```

## IV. CONFIGURACIÓN DE DJANGO CON POSTGRESQL EN DOCKER

```

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

2 DATABASES = {
3   'default': {
4     'ENGINE': 'django.db.backends.
5       postgresql_psycopg2',
6     'NAME': 'parcial2',
7     'USER': 'emerson',
8     'PASSWORD': 'kuto',
9     'HOST': 'db', # Este es el nombre del
10    servicio en Docker Compose
11    'PORT': '5432',
12  }

```

## V. COMANDOS ÚTILES DE DOCKER

### A. Construir y ejecutar servicios

```
1 docker-compose up --build
```

Este comando construye y arranca los servicios definidos en el archivo `docker-compose.yml`. La opción `--build` fuerza la reconstrucción de las imágenes antes de iniciar los contenedores.

### B. Listar contenedores activos

```
1 docker ps
```

Muestra una lista de todos los contenedores activos en tu sistema, proporcionando información como el ID del contenedor, la imagen, el comando, el tiempo de actividad y los puertos expuestos.

### C. Realizar migraciones en Django

```
1 sudo docker exec -it parcial2_docker-web-1
      python manage.py makemigrations
```

Este comando ejecuta `makemigrations` dentro del contenedor Docker especificado (`parcial2_docker-web-1`), preparando nuevas migraciones basadas en los cambios en los modelos de tu proyecto Django.

### D. Aplicar migraciones en Django

```
1 sudo docker exec -it parcial2_docker-web-1
      python manage.py migrate
```

Este comando ejecuta `migrate` dentro del contenedor Docker especificado, aplicando las migraciones pendientes y sincronizando la base de datos con los modelos actuales de tu proyecto Django.

### E. Crear un superusuario en Django

```
1 sudo docker exec -it parcial2_docker-web-1
      python manage.py createsuperuser
```

Este comando lanza el proceso interactivo de `createsuperuser` dentro del contenedor Docker especificado, permitiéndote crear un superusuario para acceder al panel de administración de Django.

## VI. RESULTADOS

```

[sudo docker — Konsola]
Archivo Editar Ver Marcadores Complementos Preferencias Ayuda
Nueva pestaña Dividir vista Copiar Pegar Buscar
>> [web internal] load build definition from Dockerfile 0.0s
--> >> transferring dockerfile: 400B 0.0s
--> [web internal] load metadata for docker.io/library/python:3.10-slim 0.0s
--> [web internal] load .dockerignore 0.0s
--> >> transferring context: 408 0.0s
--> [web 1/8] FROM docker.io/library/python:3.10-slim@sha256:eb9ca77b1a0ffbd084c1d333be3498a2638813cc25a339f8575668855b9ff1 0.0s
--> [web internal] load build context 0.0s
--> >> transferring context: 24.01B 0.0s
--> [web 2/8] RUN apt-get update && apt-get install -y git 0.0s
--> [web 3/8] RUN apt-get update && apt-get install -y libpq-dev gcc coke weasyprint 45.75
--> [web 4/8] WORKDIR /app 0.0s
--> [web 5/8] COPY requirements.txt . 0.0s
--> [web 6/8] RUN python -m pip install --upgrade pip 0.0s
--> [web 7/8] RUN python -m pip install -r requirements.txt 3.5s
--> [web 8/8] COPY . 331.0s
--> [web] exporting to image 0.1s
--> >> exporting layers 7.5s
--> >> writing image sha256:3cd5eff082b68f18d19cbf3d2927a871c6fe836fa1c7390f38ff1baef9b9c127e 0.0s
--> >> naming to docker.io/library/parcial2_docker-web 0.0s
--> [web] resolving provenance for metadata file 0.0s

```

Figura 2

Fuente: Elaboracion Propia, 2024

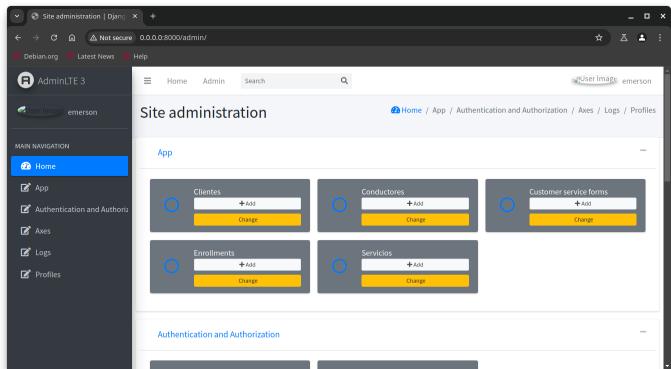


Figura 3

Fuente: Elaboracion Propia, 2024

Figura 5

Fuente: Elaboracion Propria, 2024

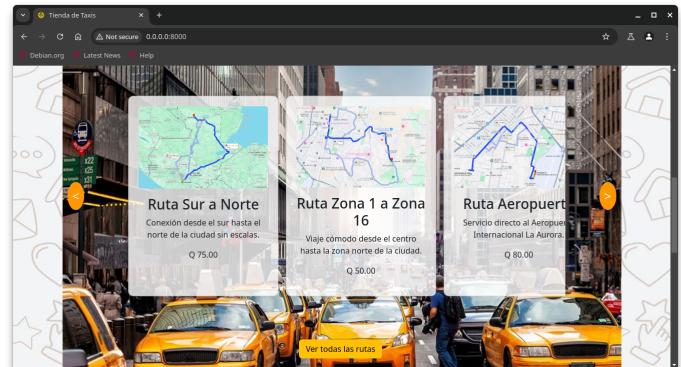


Figura 6

Fuente: Elaboracion Propria, 2024



Figura 4

Fuente: Elaboracion Propria, 2024

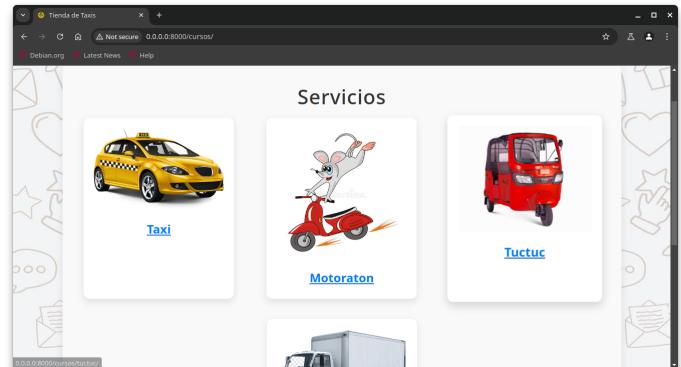


Figura 7

Fuente: Elaboracion Propria, 2024

## VII. REPOSITORIOS

Puede visitar los repositorios de los siguientes compañeros involucrados en el proyecto:

- Emerson Aldair Pérez Rivera
- Johann José Miguel Escobar González