

Universidad de San Carlos de Guatemala  
Facultad de Ingeniería  
Escuela EPS  
Prácticas Iniciales F-



# TALLER 3

# CONTENEDORES

## Grupo 8

José Leonel López Ajvix	202201211
Pablo Andres Rodriguez Lima	202201947
Franklin Orlando Noj Pérez	202200089
Carlos Alejandro Posadas Benitez	202100105
Angela Maria Esther Escobar Alvarez	202100019
Madeline Fabiola Prado Reyes	202100039

## TUTORES

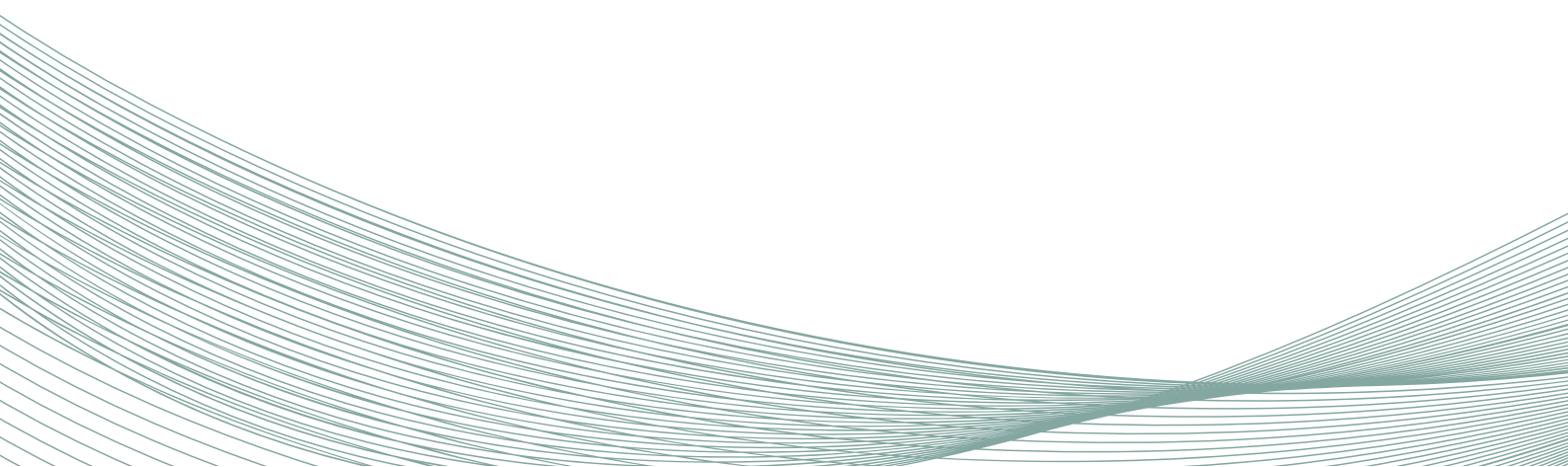
Diego André Mazariegos Barrientos	202003975
Yeinny Melissa Catalán de León	202004725



# ÍNDICE

## CONTENIDO DEL INFORME

¿Qué es Docker?	1
¿Cómo funciona?	2
Ventajas de utilizar Docker	3
Conceptos clave	4
Comandos importantes de Docker	5
Comandos utilizados en nuestro taller	5
<b>Pasos para la creación de un dockerfile</b>	6
¿Qué es un Dockerfile?	7
¿Qué es un Dockerhub?	8



# ¿QUÉ ES DOCKER?

Docker es una plataforma de virtualización de contenedores que ha transformado radicalmente la forma en que se desarrollan, implementan y administran aplicaciones en TI. Los contenedores son entornos aislados y autónomos que encapsulan aplicaciones y sus dependencias, garantizando que funcionen de manera uniforme en cualquier sistema con Docker instalado.

Esta tecnología elimina la complejidad de gestionar diferencias entre entornos de desarrollo y producción, asegurando la portabilidad y coherencia de las aplicaciones. Los contenedores Docker comparten eficientemente recursos del sistema, permitiendo la ejecución de múltiples contenedores en una sola máquina sin sobrecarga.

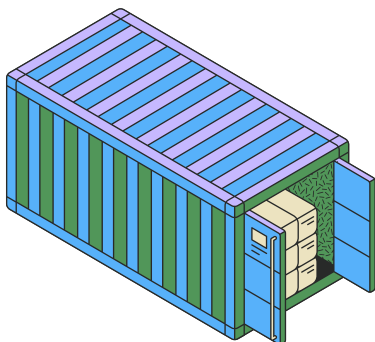
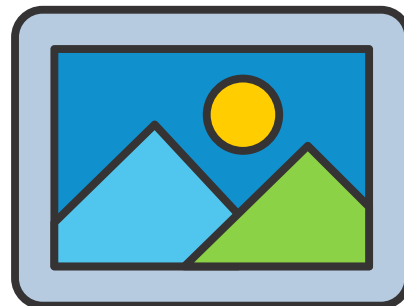
Docker se ha convertido en una herramienta esencial en el desarrollo de software y operaciones de TI, agilizando el proceso de implementación y facilitando prácticas modernas como la entrega continua. En resumen, Docker simplifica la gestión de aplicaciones y mejora la eficiencia en todo el ciclo de vida de desarrollo y operaciones.



# | ¿CÓMO FUNCIONA?

## **FUNCIONA CON IMAGENES**

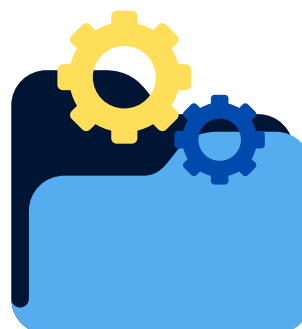
Plantillas que contienen los componentes necesarios para ejecutar una aplicación



Con las imagenes se **CREAN LOS CONTENEDORES**

Trabaja proyectos complejos en poco espacio y no importa si modificas el entorno de trabajo

**EL PROGRAMA SEGUIRA FUNCIONANDO**



Docker corre los programas sobre el

**KERNEL DEL SISTEMA OPERATIVO**

# VENTAJAS DE UTILIZAR DOCKER

## ■ PORTABILIDAD

Docker permite empaquetar aplicaciones y todas sus dependencias en contenedores, lo que garantiza que funcionen de manera consistente en cualquier entorno. Esto elimina los problemas de "funciona en mi máquina" y simplifica la gestión de aplicaciones en entornos diversos.

## ■ EFICIENCIA

Los contenedores de Docker comparten el mismo kernel del sistema operativo anfitrión, lo que los hace extremadamente eficientes en términos de recursos. Puedes ejecutar múltiples contenedores en una sola máquina sin la sobrecarga significativa que conlleva la virtualización tradicional.

## ■ AISLAMIENTO

Aunque comparten el kernel, los contenedores están completamente aislados entre sí. Cada contenedor tiene su propio sistema de archivos y espacio de nombres de procesos, lo que garantiza la seguridad y la separación de aplicaciones.

## ■ ESCALABILIDAD

Docker facilita la escalabilidad de las aplicaciones. Puedes escalar contenedores horizontalmente (agregando más instancias) o verticalmente (aumentando los recursos) según las necesidades de tu aplicación.

## ■ FLEXIBILIDAD

Puedes utilizar Docker para empaquetar una variedad de aplicaciones, desde aplicaciones web hasta bases de datos, lo que lo hace versátil y adecuado para una amplia gama de casos de uso.



# CONCEPTOS CLAVE

Para comprender Docker de manera efectiva, es esencial estar familiarizado con algunos conceptos clave relacionados con esta tecnología de contenedores. Aquí tienes los conceptos más importantes:

- **Contenedor:** Una unidad de ejecución aislada que incluye una aplicación y todas sus dependencias, permitiendo una ejecución consistente en cualquier entorno.
- **Imagen:** Una plantilla de solo lectura que contiene todo lo necesario para ejecutar una aplicación, como archivos, bibliotecas y configuraciones.
- **Dockerfile:** Un archivo de texto que describe cómo construir una imagen de Docker, especificando la configuración y la instalación de software.
- **Docker Hub:** Un repositorio en línea para compartir y descargar imágenes de Docker preconfiguradas y listas para usar.
- **Docker Compose:** Una herramienta para definir y gestionar aplicaciones multi-contenedor, facilitando la gestión de aplicaciones complejas.
- **Orquestación de Contenedores:** La gestión de múltiples contenedores que trabajan juntos como parte de una aplicación, facilitada por herramientas como Docker Swarm o Kubernetes.
- **Daemon de Docker:** Un servicio en segundo plano que administra operaciones de Docker en el sistema operativo del servidor.
- **CLI de Docker:** La interfaz de línea de comandos para interactuar con Docker, utilizada para crear, gestionar y controlar contenedores e imágenes.
- **Volúmenes:** Mecanismos de almacenamiento que permiten a los contenedores acceder y persistir datos más allá de su vida útil.
- **Redes de Docker:** Redes virtuales que permiten la comunicación entre contenedores y con el exterior, esenciales para la interacción y exposición de servicios.

## COMANDOS IMPORTANTES DE DOCKER

- **docker --version:** Verificar la versión de Docker
- **docker pull:** Descargar una imagen desde DockerHub
- **docker images:** Listar imágenes locales
- **docker run:** Ejecutar un contenedor
- **docker ps:** Listar contenedores en ejecución
- **docker stop y docker start:** Detener e iniciar contenedores
- **docker rm:** Eliminar contenedores
- **docker rmi:** Eliminar imágenes
- **docker logs:** Ver registros de un contenedor
- **docker exec:** Ejecutar comandos en un contenedor en ejecución

## COMANDOS UTILIZADOS EN NUESTRO TALLER

### PASOS PARA LA CREACIÓN DE UN DOCKERFILE

#### 1. Creamos nuestro dockerfile



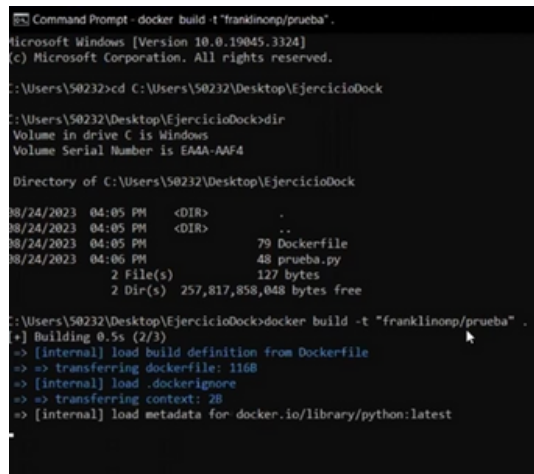
```
File Edit Format View Help
FROM python:latest

COPY . \app

WORKDIR \app

CMD ["python", "prueba.py"]
```

2. Para saber que tiene adentro nuestro dockerfile desde la consola escribimos `cd "seguido de la ruta de nuestro archivo"`. Luego escribimos `dir` y sabremos lo que hay dentro de la carpeta.



```
Command Prompt - docker build -t "franklinop/prueba".
Microsoft Windows [Version 10.0.19045.3124]
(c) Microsoft Corporation. All rights reserved.

C:\Users\S0232>cd C:\Users\S0232\Desktop\EjercicioDock

C:\Users\S0232\Desktop\EjercicioDock>dir
Volume in drive C is Windows
Volume Serial Number is EAAA-AAF4

Directory of C:\Users\S0232\Desktop\EjercicioDock

08/24/2023  04:05 PM    <DIR>          .
08/24/2023  04:05 PM    <DIR>          ..
08/24/2023  04:05 PM                79 Dockerfile
08/24/2023  04:06 PM                48 prueba.py
                2 File(s)            127 bytes
                2 Dir(s)  257,817,858,048 bytes free

C:\Users\S0232\Desktop\EjercicioDock>docker build -t "franklinop/prueba" .
[*] Building 0.5s (2/3)
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 116B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/python:latest
```

3. Docker build -t "franklinonp/prueba".

4. Docker images para saber que imagenes hay dentro.

```

Command Prompt
> [auth] library/python:pull token for registry-1.docker.io
> [internal] load build context
> > transferring context: 202B
> CACHED [1/3] FROM docker.io/library/python:latest@sha256:85b3d192ddbc96588b719e86991e472b390805a754681a38132
> [2/3] COPY . app
> [3/3] ADD src018 app
> exporting to image
> > exporting layers
> > writing image sha256:9c8b97da98bb0d32be8fdaa90d0c60f8a6cc10d92df5b9884adeeaeedbe5aed
> > naming to docker.io/franklinonp/prueba
What's Next?
View summary of image vulnerabilities and recommendations + docker scout quickview
C:\Users\S0232\Desktop\EjercicioDocker> docker images
REPOSITORY          TAG         IMAGE ID      CREATED       SIZE
franklinonp/prueba  latest      9c8b97da98bb  15 seconds ago  1.01GB
cnone>              cnone>      7a92f1de9153  15 minutes ago  1.01GB
cnone>              cnone>      2cdcd2e6e1a5  About an hour ago  1.01GB
C:\Users\S0232\Desktop\EjercicioDocker> docker run franklinonp/prueba
Hola compañeros y tutores del grupo #8
C:\Users\S0232\Desktop\EjercicioDocker>

```

5. Docker run frnklinonp/prueba ejecutara nuestra prueba.

6. Para descargar la imagen de prueba docker run -d -p 80:80 docker/getting-started.

```

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

pablo03r@LAPTOP-JUHIOL5G:~$ docker run -d -p 80:80 docker/getting-started
Unable to find image 'docker/getting-started:latest' locally
latest: Pulling from docker/getting-started
c158987b0551: Downloading [=====]
1e35f6679fab: Download complete
cb9626c74200: Download complete
b6334b6ace34: Download complete
f1d1c9928c82: Download complete
9b6f639ec6ea: Download complete
ee68d3549ec8: Downloading [=====]
33e0cbbb4673: Download complete
4f7e34c2de10: Waiting

```

7. Descargar imagen docker pull hello-world.

8. Docker pull mongo para subir nuestra imagen.

9. Ya cargada nuestra base de datos ejecutamos el comando docker --name taller3g8 -p 4200:4200 taller3gr8:latest.

10. En el backend colocamos docker build -t api8.

11. Ya cargada la imagen colocamos docker run --name api8 -p 3000:3000 api8.

12. Docker login

13. Docker tag api8:latest direccion de nuestro dockerfile

14. Docker push "direccion del dockerfile"



## ¿QUE ES UN DOCKERFILE?

Un Dockerfile es un archivo de texto plano que contiene instrucciones para crear una imagen de Docker. Una imagen de Docker es un paquete que incluye una aplicación y todas sus dependencias, junto con información sobre cómo ejecutar la aplicación. Los Dockerfiles son esenciales para la construcción de imágenes personalizadas en Docker y permiten la reproducibilidad y la automatización del proceso de construcción de contenedores.

### ESTRUCTURA DE UN DOCKERFILE

Un Dockerfile generalmente sigue una estructura básica que incluye una serie de instrucciones. Algunas de las instrucciones más comunes incluyen:

1. **FROM:** Esta instrucción especifica la imagen base a partir de la cual se construirá la nueva imagen. Por ejemplo, `FROM ubuntu:20.04` establecerá una imagen base de Ubuntu 20.04.
2. **RUN:** Utilizada para ejecutar comandos dentro del contenedor durante la construcción de la imagen. Por ejemplo, `RUN apt-get update && apt-get install -y nginx` instalará el servidor web Nginx en la imagen.
3. **COPY y ADD:** Estas instrucciones se utilizan para copiar archivos desde el sistema de archivos del host al sistema de archivos del contenedor. Por ejemplo, `COPY app.py /app/` copiará el archivo "app.py" desde el directorio actual del host al directorio "/app/" en el contenedor.
4. **WORKDIR:** Establece el directorio de trabajo en el contenedor para las instrucciones futuras. Por ejemplo, `WORKDIR /app` cambiará el directorio de trabajo actual al directorio "/app".
5. **EXPOSE:** Esta instrucción especifica el puerto en el que el contenedor escuchará las conexiones entrantes. Por ejemplo, `EXPOSE 80` expone el puerto 80.
6. **CMD y ENTRYPOINT:** Se utilizan para especificar el comando que se ejecutará cuando se inicie el contenedor. `CMD` proporciona argumentos predeterminados para el comando, mientras que `ENTRYPOINT` especifica el comando principal.

## ¿QUE ES UN DOCKERHUB?

Docker Hub es un servicio en la nube proporcionado por Docker, Inc. que actúa como un registro de contenedores. Permite a los desarrolladores, equipos y organizaciones almacenar, distribuir y compartir imágenes de Docker públicas y privadas. Es una plataforma central para encontrar y distribuir imágenes de contenedores, lo que facilita la creación, distribución y administración de aplicaciones basadas en Docker.

- **Descarga de imágenes desde Docker Hub:**

Para descargar una imagen de Docker Hub, simplemente utiliza el comando **docker pull** seguido del nombre de la imagen que deseas descargar. Por ejemplo: **docker pull nombreusuario/mi-imagen:1.0**.

- **Etiquetas y versiones en Docker Hub:**

Docker Hub permite etiquetar y versionar las imágenes de contenedores. Las etiquetas son versiones específicas de una imagen. Por ejemplo, puedes tener una imagen llamada "mi-aplicacion" con etiquetas como "1.0", "latest", "testing", etc. Para descargar una versión específica de una imagen, debes incluir la etiqueta en el nombre de la imagen, por ejemplo, **nombreusuario/mi-aplicacion:1.0**.

- **Colaboración y control de versiones en Docker Hub:**

Docker Hub ofrece funcionalidades de colaboración y control de versiones para imágenes de contenedores. Puedes invitar a otros usuarios a colaborar en tus repositorios de imágenes, establecer políticas de acceso y controlar quién puede ver y modificar tus imágenes. Además, puedes configurar la automatización de la construcción de imágenes cada vez que se actualiza tu código fuente o cuando se modifican archivos en tu repositorio de origen.