

◆ Ejercicio 1 - Manual Docker

📌 Manual Paso a Paso para el Uso de Docker Desktop

🚀 Introducción

Docker Desktop es una aplicación que permite la gestión y ejecución de contenedores Docker en sistemas operativos Windows y macOS. Este manual proporciona una guía paso a paso para instalar, configurar y utilizar Docker Desktop de manera eficiente.

✓ Requisitos Previos

Antes de instalar Docker Desktop, asegúrate de cumplir con los siguientes requisitos:

- ◆ **Windows:** Windows 10 (versión 1903 o superior) o Windows 11 con WSL 2 habilitado.
- ◆ **Mac:** macOS 11 (Big Sur) o superior con Apple Silicon o procesador Intel.
- ◆ **Memoria RAM:** Mínimo **4 GB** recomendados.
- ◆ **Espacio en Disco:** Al menos **10 GB** de espacio libre.
- ◆ **Conexión a Internet:** Requerida para la descarga e instalación.

Instalación de Docker Desktop

Descargar Docker Desktop

- 1 Ve al sitio oficial de Docker: [Docker Desktop](#)
- 2 Descarga la versión correspondiente a tu sistema operativo (Windows o macOS).

Instalar Docker Desktop

Windows:

- 1 Ejecuta el archivo **.exe** descargado.
- 2 Selecciona la opción para habilitar **WSL 2** (si es compatible con tu sistema).
- 3 Sigue las instrucciones del instalador y **reinicia** tu equipo si es necesario.

Mac:

- 1 Abre el archivo **.dmg** descargado.
- 2 Arrastra el ícono de **Docker** a la carpeta **Aplicaciones**.
- 3 Abre Docker Desktop desde **Aplicaciones** y sigue las instrucciones de configuración inicial.

Configuración Inicial

- Abre **Docker Desktop** tras la instalación.
- Acepta los términos y condiciones.
- (Opcional) Configura el uso de recursos en "Settings" (**CPU, memoria y disco**).
- Asegúrate de que **Docker Engine** esté corriendo (**verás el ícono de Docker en la barra de tareas o menú superior**).

Uso Básico de Docker

Verificar la Instalación

Abre una terminal (**PowerShell en Windows, Terminal en macOS**) y ejecuta:

```
docker --version
```

Si ves un número de versión, significa que **Docker está instalado correctamente.** ✓

⌚ Ejecutar un Contenedor de Prueba

Para comprobar que Docker funciona, ejecuta:

```
docker run hello-world
```

🔊 Esto descargará y ejecutará un contenedor de prueba, mostrando un mensaje de confirmación.

📷 Imagen:

```
PS C:\Windows\system32> docker --version
Docker version 24.0.7, build afdd53b
PS C:\Windows\system32> docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
e6590344b1a5: Pull complete
Digest: sha256:e0b569a5163a5e6be84e210a2587e7d447e08f87a0e90798363fa44a0464a1e8
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/
```

📦 Descargar y Ejecutar una Imagen de Docker

Ejemplo con **Nginx**:

```
docker pull nginx
```

Para ejecutar el contenedor:

```
docker run -d -p 8080:80 nginx
```

 Accede a <http://localhost:8080> para ver el servidor web en ejecución.

 **Imagen:**

```
PS C:\Windows\system32> docker pull nginx
Using default tag: latest
latest: Pulling from library/nginx
c29f5b76f736: Pull complete
e19db8451adb: Pull complete
24ff42a0d907: Pull complete
c558df217949: Pull complete
976e8f6b25dd: Pull complete
6c78b0ba1a32: Pull complete
84cade77a831: Pull complete
Digest: sha256:91734281c0ebfc6f1aea979cffeed5079cfe786228a71cc6f1f46a228cde6e34
Status: Downloaded newer image for nginx:latest
docker.io/library/nginx:latest
[PS C:\Windows\system32> docker run -d -p 8080:80 nginx
40415e149526146fe3b9abe079166e35538296108e2ae45142619257daab2d8b
```

 **Listar Contenedores Activos**

```
docker ps
```

Para ver **todos** los contenedores, incluyendo los detenidos:

```
docker ps -a
```

 **Imagen:**

```
PS C:\Windows\system32> docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS          PORTS          NAMES
40415e149526   nginx      "/docker-entrypoint..."   3 minutes ago  Up 3 minutes   0.0.0.0:8080->80/tcp   jovial_rhodes
PS C:\Windows\system32> docker ps -a
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS          PORTS          NAMES
40415e149526   nginx      "/docker-entrypoint..."   3 minutes ago  Up 3 minutes   0.0.0.0:8080->80/tcp   jovial_rhodes
64ef6e5ab960   hello-world "/hello"                7 minutes ago  Exited (0) 7 minutes ago
033be259ec9d   mysql:5.7  "docker-entrypoint.s..."  13 months ago  Exited (0) 2 months ago
                                         mysqlAD2
```

Detener y Eliminar Contenedores

Para **detener** un contenedor:

```
docker stop <ID_DEL_CONTENEDOR>
```

Para **eliminarlo**:

```
docker rm <ID_DEL_CONTENEDOR>
```

Imagen:

```
PS C:\Windows\system32> docker stop 40415e149526
40415e149526
PS C:\Windows\system32> docker rm 40415e149526
40415e149526
```

Configuración Avanzada

Configurar Volúmenes

Para persistir datos entre reinicios de contenedores:

```
docker volume create mi-volumen
```

Para montar el volumen en un contenedor:

```
docker run -d -v mi-volumen:/app/data nginx
```

Imagen:

```
PS C:\Windows\system32> docker volume create mi-volumen  
mi-volumen  
PS C:\Windows\system32> docker run -d -v mi-volumen:/app/data nginx  
0006b9105608dd7fac8c955a2a72c0e46db5c6a926926103c3aad608e632c4c7
```

Configurar Redes Personalizadas

Para crear una red personalizada:

```
docker network create mi-red
```

Para conectar un contenedor a la red:

```
docker run -d --network mi-red nginx
```

Imagen:

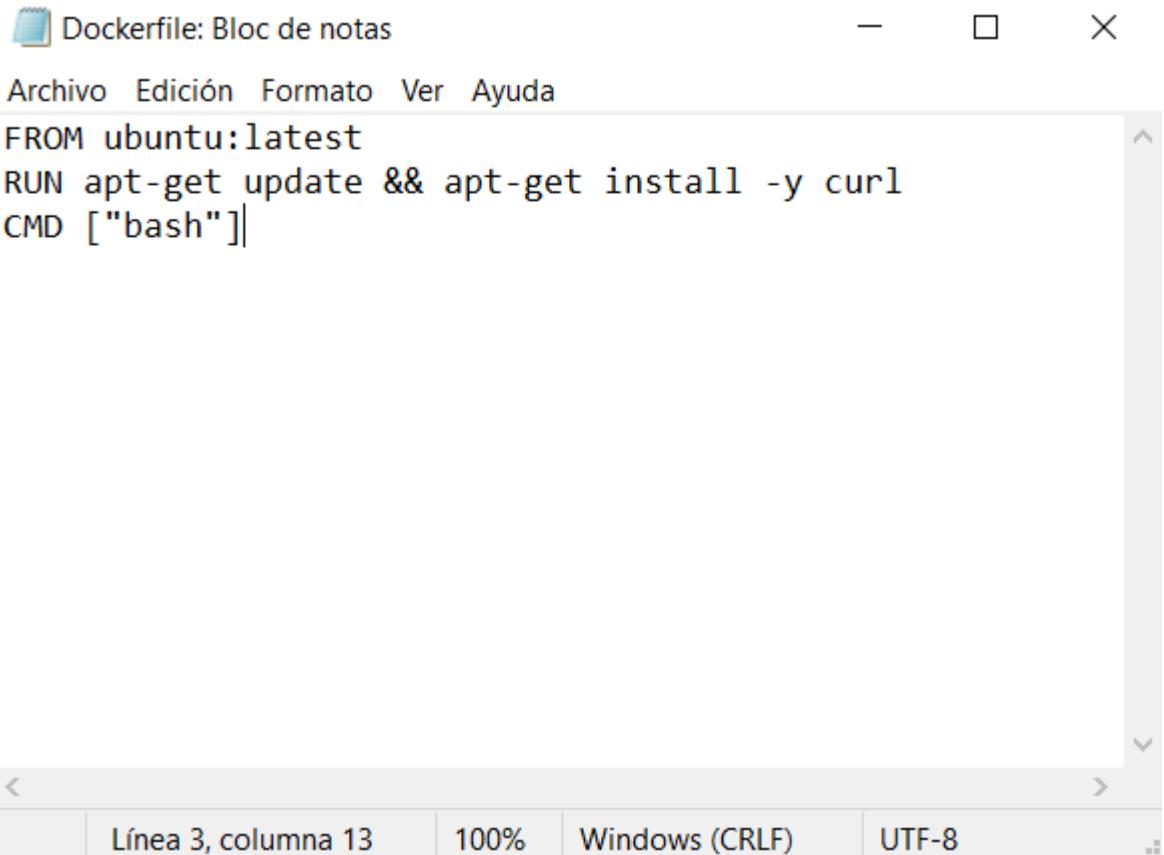
```
PS C:\Windows\system32> docker network create mi-red  
2ad5cee9ca825387412fbb902d0fc9102ae121cd97ec09f264cc927ca36cc8f  
PS C:\Windows\system32> docker run -d --network mi-red nginx  
63e643f1a0f049bce57749702ecec996567994fb640ac7d0d376565efc1d7222
```

Construir una Imagen Personalizada

Crea un archivo `Dockerfile` con el siguiente contenido:

```
FROM ubuntu:latest  
RUN apt-get update && apt-get install -y curl  
CMD ["bash"]
```

Imagen:



Dockerfile: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
FROM ubuntu:latest
RUN apt-get update && apt-get install -y curl
CMD ["bash"]
```

Línea 3, columna 13 | 100% | Windows (CRLF) | UTF-8

Construye la imagen con:

```
docker build -t mi-imagen .
```

Ejecuta un contenedor basado en la imagen:

```
docker run -it mi-imagen
```

 **Imagen:**

```
PS C:\Users\diego\Desktop\Docker-Desktop\mi-imagen> docker build -t mi-imagen .
[+] Building 20.8s (6/6) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 116B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/ubuntu:latest
=> [1/2] FROM docker.io/library/ubuntu:latest
=> [2/2] RUN apt-get update && apt-get install -y curl
=> exporting to image
=> => exporting layers
=> => writing image sha256:a57aac6d7e912c8cc7a8ce701374059ac6d65d4787b21c47d60ffd71c9c01012
=> => naming to docker.io/library/mi-imagen
```

What's Next?

View a summary of image vulnerabilities and recommendations → `docker scout quickview`
PS C:\Users\diego\Desktop\Docker-Desktop\mi-imagen> docker run -it mi-imagen
root@762a2c95a035:/#

 **Responsables:** @Diego Suárez