



Tarea Evaluable Docker

CIFP La Laboral - Módulo Despliegue de Aplicaciones Web

 **Autores:** Iker y Pelayo Rodríguez

 **Fecha de entrega y exposición:** Viernes, 21 de febrero de 2025.

 **Repositorio GitHub:**

ÍNDICE

| | |
|---|--|
|  | 1. Descripción del Proyecto |
|  | 2. Instrucciones Generales |
|  | 3. Asignación de tareas |
|  | 4. Estructura del Repositorio |
|  | 5. Ejercicios |
| | 5.1 Manual de Docker Desktop |
| | 5.2 Servidor de Base de Datos |
| | 5.3 Contenedores en Red |
| | 5.4 Docker Compose |
| | 5.5 Imagen con Dockerfile - Aplicación Web |



Descripción del Proyecto

Este repositorio contiene la resolución de la **Tarea Evaluable Docker**, organizada en cinco ejercicios prácticos relacionados con la gestión de contenedores Docker, redes y despliegue de aplicaciones web.

Instrucciones Generales

- **Trabajo en equipo:** Grupo compuesto por 2 personas(Pelayo e Iker).
- **Repositorio en GitHub:** Tenemos una persona que creó el repositorio(Pelayo) y otra colaboradora(Iker).
- **Uso de ramas:** Trabajamos con branches para una correcta organización del código.
- **Registro de contribuciones:** Los commits reflejan las aportaciones de cada miembro del equipo.
- **Asignación de tareas:** Documentación de la distribución de tareas, plazos y porcentaje de cumplimiento en un documento del repositorio.
- **Estructura del repositorio:** Contamos con una carpeta para cada ejercicio con la documentación y los archivos necesarios.
- **Entrega final:** Documento en PDF con nuestros nombres y la URL del repositorio.

ASIGNACIÓN DE TAREAS

ToDo

| Aa Nombre | Asignar | Ejercicio 1 | Ejercicio 2 | Ejercicio 3 | Ejercicio 4 | Ejercicio 5 |
|--------------------------------|--------------------|-------------|-------------|-------------|-------------|-------------|
| <u>Ejercicios</u> | Iker Pérez Pelayo | Listo | Listo | Listo | Listo | Listo |
| <u>Exportación de archivos</u> | Iker Pérez Pelayo | Listo | Listo | Listo | Listo | Listo |
| <u>Repository</u> | Iker Pérez Pelayo | Listo | Listo | Listo | Listo | Listo |
| <u>Carpetas de Ejercicios</u> | | Listo | Listo | Listo | Listo | Listo |
| <u>Project (ToDo)</u> | Iker Pérez | Listo | Listo | Listo | Listo | Listo |

📁 ESTRUCTURA DEL REPOSITORIO

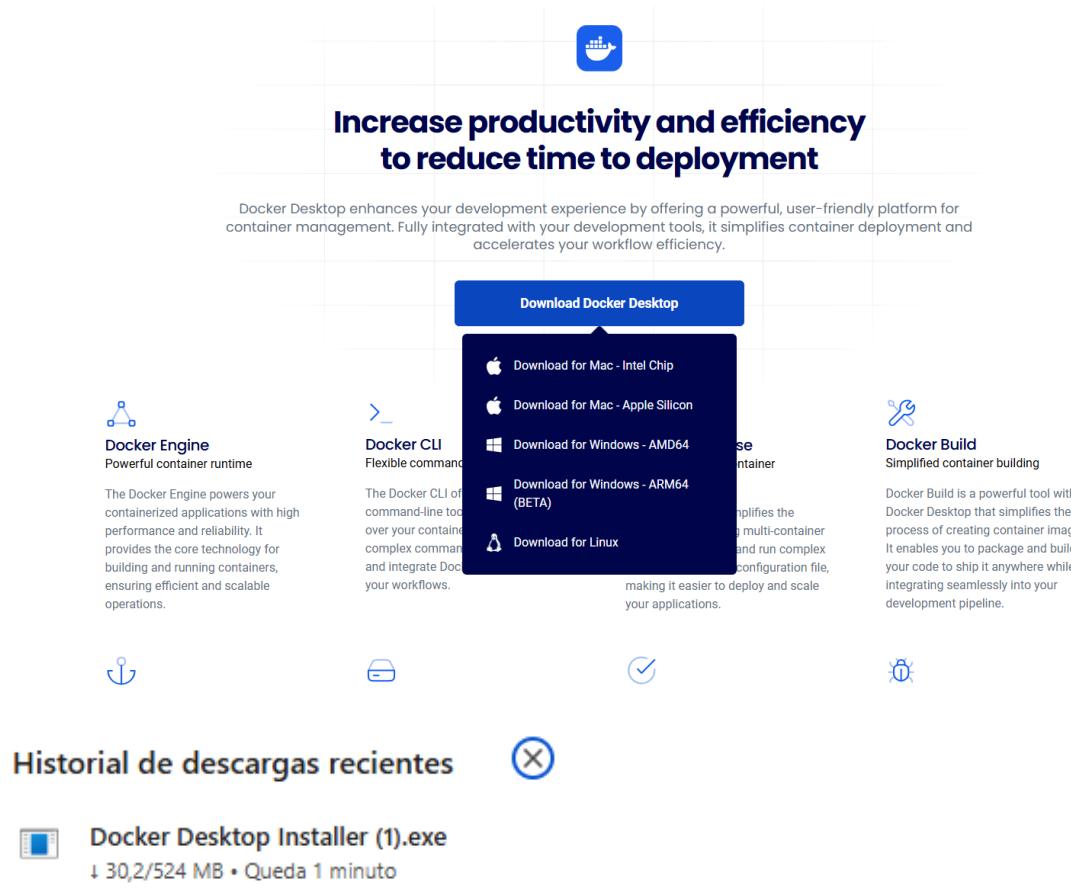
```
 proyecto-docker/
    ├── Ejercicio_1/
    │   ├── manual_docker.md
    │   └── imagenes/
    │       └── manual_docker.pdf
    ├── Ejercicio_2/
    │   ├── servidor_base_datos.md
    │   └── imagenes/
    │       └── servidor_base_datos.pdf
    ├── Ejercicio_3/
    │   ├── comandos_red.md
    │   └── imagenes/
    │       └── comandos_red.pdf
    ├── Ejercicio_4/
    │   ├── docker_compose.yaml
    │   └── imagenes/
    │       ├── docker_compose.md
    │       └── docker_compose.pdf
    ├── Ejercicio_5/
    │   ├── Dockerfile
    │   ├── index.html
    │   ├── styles.css
    │   ├── script.php
    │   └── imagenes/
    │       ├── docker_commands.txt
    │       ├── dockerfile.md
    │       └── dockerfile.pdf
    └── README.md
```

Ejercicios

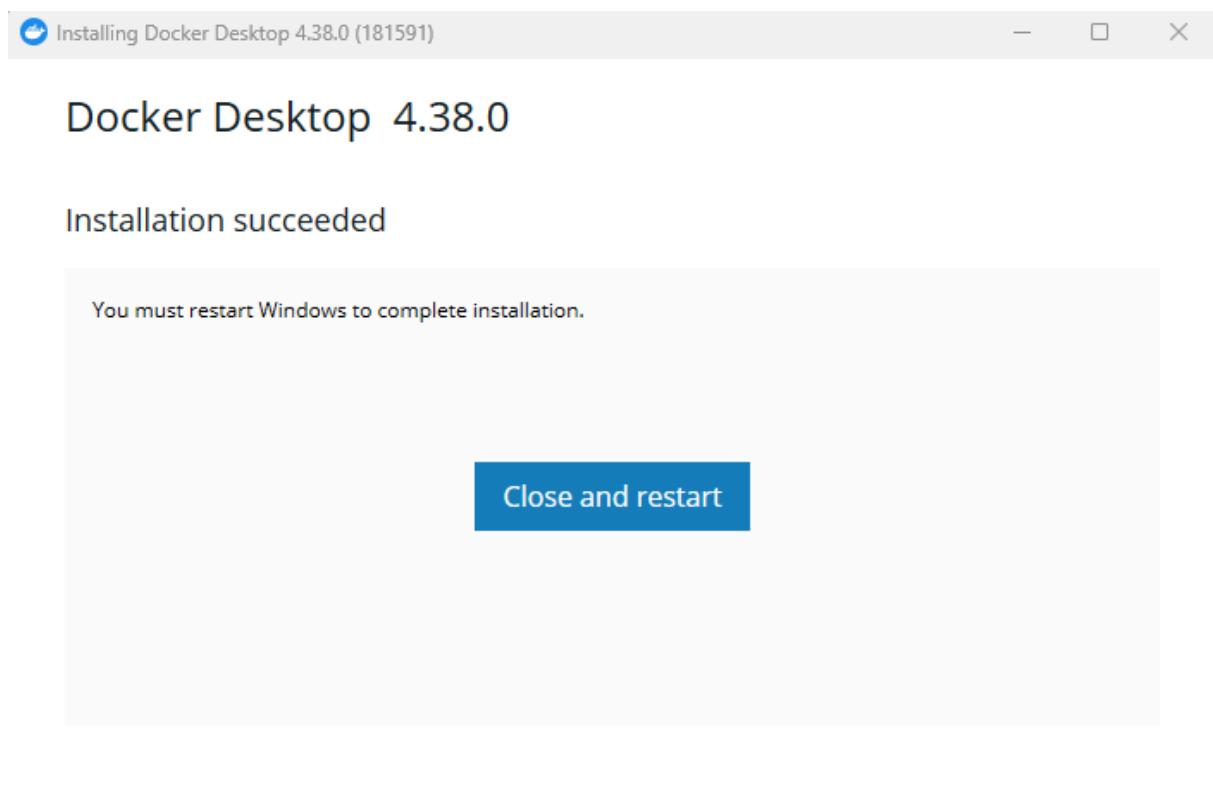
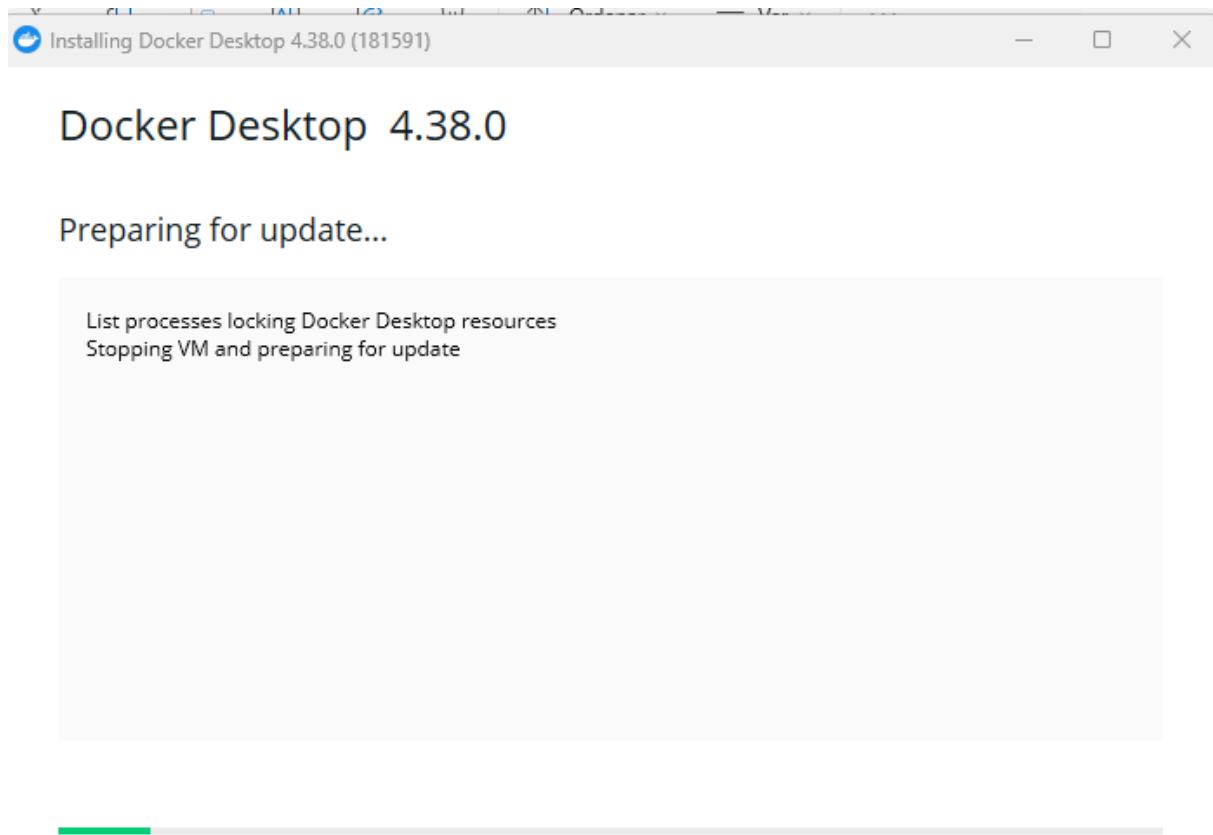
Ejercicio 1: Manual de Docker Desktop

Proceso paso a paso de instalación

- Abrimos el navegador y accedemos a la página oficial de Docker:
 [Página oficial de Docker Desktop](#)
- Bajamos un poco pinchamos en Download Docker Desktop y lo descargamos para Windows.

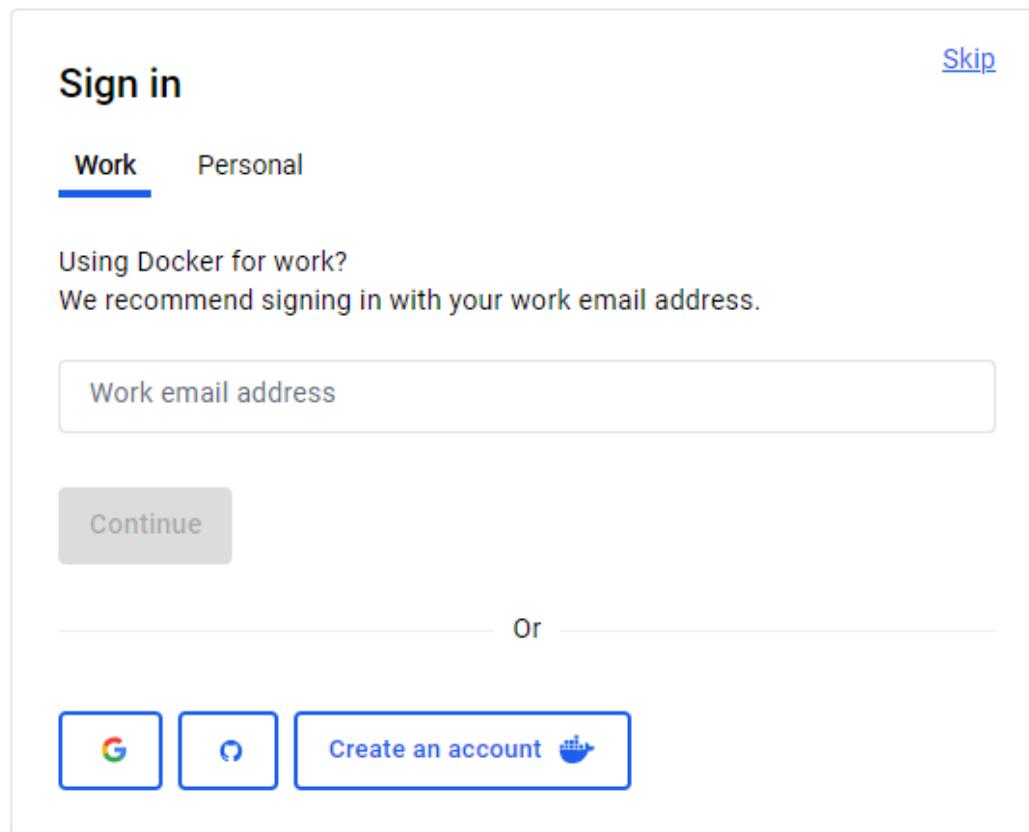


Una vez descargado, pinchamos en el instalador y comenzamos con la descarga:



Una vez realizada la instalación, accedemos a Docker Desktop:

Nada mas entrar nos pedirá iniciar sesión o registrarnos:



Una vez hecho, ya accedemos a Docker Desktop y a todas sus funcionalidades:

| Name | Container ID | Image | Port(s) | CPU (%) | Last started | Actions |
|-----------------|--------------|--------------|---------|---------|--------------|---------|
| happy_engelbart | 198663c274e2 | nginx:latest | - | N/A | 2 days ago | |
| angry_moser | 743da717df07 | hello-world | - | N/A | 2 days ago | |
| httpd-docker | - | - | - | N/A | - | |

💻 Navegación por la interfaz principal



Containers Give feedback

View all your running containers and applications. [Learn more](#)

Container CPU usage (1) Container memory usage (1)

No containers are running.

Search Only show running containers

| Name | Image | Status | Port(s) | CPU (%) | Last started | Actions |
|-----------------------------|-------------|--------|---------|------------|--------------|---------|
| angry_moser 743da717df07 | hello-world | Exited | N/A | 1 hour ago | | |

[Show charts](#)

Images Give feedback

View and manage your local and Docker Hub images. [Learn more](#)

Local Hub repositories

10.07 KB / 0 Bytes in use 1 images Last refresh: 1 hour ago

Search

| Name | Tag | Status | Created | Size | Actions |
|-----------------------------|--------|--------|-------------|----------|---------|
| hello-world 74cc54e270c4 | latest | In use | 21 days ago | 10.07 KB | |

Volumes Give feedback

Manage your volumes, view usage, and inspect their contents. [Learn more](#)

Containers can use volumes to store data
All data in a container is lost once it is removed. Containers use volumes to persist data.

[Create a volume](#)

Builds Give feedback

Build container images and artifacts from source code. [Learn more](#)

Selected builder: desktop-linux [Import builds](#) [Builder settings](#)

Build history Active builds

Search Show only my builds

| ID | Name | Builder | Status | Duration | Created | Author |
|----|-----------------------|---|--------|----------|---------|--------|
| | No build record found | If you can't see your builds here make sure your builder Instances are running. | | | | |

Operaciones básicas con contenedores

Ver Contenedores Disponibles

- Abrimos **Docker Desktop**.
- En la barra lateral izquierda, seleccionamos **Containers**.
- Se muestra una lista con los contenedores creados, indicando su estado (corriendo, detenido, etc.).

The screenshot shows the Docker interface with a search bar and a filter for 'Only show running containers'. A table lists one container:

| Name | Image | Status | Port(s) |
|-------------|-------------|--------|---------|
| angry_moser | hello-world | Exited | |

A context menu is open over the container row, showing options like 'View details', 'View image packages and CVEs', 'Copy docker run', 'Open in terminal', 'View files', 'Pause', 'Restart', and 'Delete'.

Iniciar un Contenedor Detenido

- Ubicamos el contenedor en la lista.
- Si el estado es "Exited", hacemos clic en los tres puntos (:) a la derecha del contenedor.
- Seleccionamos Restart para reiniciarlo.

The container details page for 'angry_moser' shows the following information:

- Logs:** Displays the logs of the container, which include a welcome message from Docker.
- STATUS:** Exited (0) (13 minutes ago).
- Actions:** Buttons for Start, Stop, and Restart.

```

2025-02-12 10:13:19 Hello from Docker!
2025-02-12 10:13:19 This message shows that your installation appears to be working correctly.
2025-02-12 10:13:19
2025-02-12 10:13:19 To generate this message, Docker took the following steps:
2025-02-12 10:13:19 1. The Docker client contacted the Docker daemon.
2025-02-12 10:13:19 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
2025-02-12 10:13:19 (amd64)
2025-02-12 10:13:19 3. The Docker daemon created a new container from that image which runs the
2025-02-12 10:13:19 executable that produces the output you are currently reading.
2025-02-12 10:13:19 4. The Docker daemon streamed that output to the Docker client, which sent it
2025-02-12 10:13:19 to your terminal.
2025-02-12 10:13:19
2025-02-12 10:13:19 To try something more ambitious, you can run an Ubuntu container with:
2025-02-12 10:13:19 $ docker run -it ubuntu bash
2025-02-12 10:13:19
2025-02-12 10:13:19 Share images, automate workflows, and more with a free Docker ID:
2025-02-12 10:13:19 https://hub.docker.com/
2025-02-12 10:13:19
2025-02-12 10:13:19 For more examples and ideas, visit:
2025-02-12 10:13:19 https://docs.docker.com/get-started/
2025-02-12 10:13:19

```

Detener un Contenedor en Ejecución

- Usamos el contenedor en ejecución.
- Le damos a los tres puntos (:).
- Selecciona **Pause** o **Stop** según sea necesario.

The screenshot shows the Docker interface with a search bar and a filter for 'Only show running containers'. A table lists one container:

| Name | Image | Status | Port(s) | CPU (%) | Last started | Start | Logs |
|-------------|-------------|--------|---------|---------|----------------|-------|------|
| angry_moser | hello-world | Exited | | N/A | 45 seconds ago | | |

Eliminar un Contenedor

- Nos aseguramos de que el contenedor está detenido.
- Hacemos clic en los tres puntos (:) a la derecha del contenedor.
- Seleccionamos el icono de Eliminar para eliminarlo.

| Name | Image | Status | Port(s) | CPU (%) | Last started | Actions |
|-----------------------------|-------------|--------|---------|---------|----------------|--|
| angry_moser 743da717df07 | hello-world | Exited | | N/A | 16 minutes ago | More ⋮ Delete |

Ver Detalles de un Contenedor

- Hacemos clic en los tres puntos (:) a la derecha del contenedor.
- Seleccionamos view details.
- Al entrar, podemos ver información como logs, configuración y estadísticas de uso.

| Name | Image | Status | Port(s) |
|-----------------------------|-------------|--------|---------|
| angry_moser 743da717df07 | hello-world | Exited | |

| Logs | Inspect | Bind mounts | Exec | Files | Stats |
|---|---------|-------------|------|-------|-------|
| <pre>2025-02-12 10:13:00 Hello from Docker! 2025-02-12 10:13:00 This message shows that your installation appears to be working correctly. 2025-02-12 10:13:00 2025-02-12 10:13:00 To generate this message, Docker took the following steps: 2025-02-12 10:13:00 1. The Docker client contacted the Docker daemon. 2025-02-12 10:13:00 2. The Docker daemon pulled the "hello-world" image from the Docker Hub. 2025-02-12 10:13:00 (amd64) 2025-02-12 10:13:00 3. The Docker daemon created a new container from that image which runs the 2025-02-12 10:13:00 executable that produces the output you are currently reading. 2025-02-12 10:13:00 4. The Docker daemon streamed that output to the Docker client, which sent it 2025-02-12 10:13:00 to your terminal. 2025-02-12 10:13:00 2025-02-12 10:13:00 To try something more ambitious, you can run an Ubuntu container with: 2025-02-12 10:13:00 \$ docker run -it ubuntu bash 2025-02-12 10:13:00 2025-02-12 10:13:00 Share images, automate workflows, and more with a free Docker ID: 2025-02-12 10:13:00 https://hub.docker.com/ 2025-02-12 10:13:00 2025-02-12 10:13:00 For more examples and ideas, visit: 2025-02-12 10:13:00 https://docs.docker.com/get-started/ 2025-02-12 10:13:10</pre> | | | | | |

```

angry_moser
743da717df07 ⚡ hello-world:latest
Logs Inspect Bind mounts Exec Files Stats
Platform Cmd State Image PortBindings Runtime Mounts Volumes Env Labels Networks
1 v {
2   "Id": "743da717df07c605f01ef26d7ae3263d812eeffa31676235b527500f27ba38d32",
3   "Created": "2025-02-12T08:00:04.835224825Z",
4   "Path": "/hello",
5   "Args": [],
6   "State": {
7     "Status": "exited",
8     "Running": false,
9     "Paused": false,
10    "Restarting": false,
11    "OOMKilled": false,
12    "Dead": false,
13    "Pid": 0,
14    "ExitCode": 0,
15    "Error": "",
16    "StartedAt": "2025-02-12T09:13:19.031056818Z",
17    "FinishedAt": "2025-02-12T09:13:19.248875412Z"
18  },
19  "Image": "sha256:74c54e27dc1bb10d4b2226072d469509f2f222fa13ce1454a59661a1d44602",
20  "ResolvConfPath": "/var/lib/docker/containers/743da717df07c605f01ef26d7ae3263d812eeffa31676235b527500f27ba38d32/resolv.conf",
21  "HostnamePath": "/var/lib/docker/containers/743da717df07c605f01ef26d7ae3263d812eeffa31676235b527500f27ba38d32/hostname",
22  "HostsPath": "/var/lib/docker/containers/743da717df07c605f01ef26d7ae3263d812eeffa31676235b527500f27ba38d32/hosts",
23  "LogPath": "/var/lib/docker/containers/743da717df07c605f01ef26d7ae3263d812eeffa31676235b527500f27ba38d32/logs",
24  "Name": "angry_moser",
25  "RestartCount": 0,
26  "Driver": "overlay2",
27  "Platform": "linux",
28  "MountLabel": "",
29  "ProcessLabel": "",
30  "AppArmorProfile": "",
31  "ExecIDs": null,
32  "HostConfig": {
33    "Binds": null,
34    "ContainerIDFile": "",
35    "LogConfig": {
36      "Type": "json-file",
37      "Config": {}
38    }
39  }
40}

```

Copiar el Comando de Ejecución de un Contenedor

- Hacemos clic en los tres puntos (:) junto al contenedor.
- Seleccionamos **Copy docker run** para copiar el comando con el que fue creado.

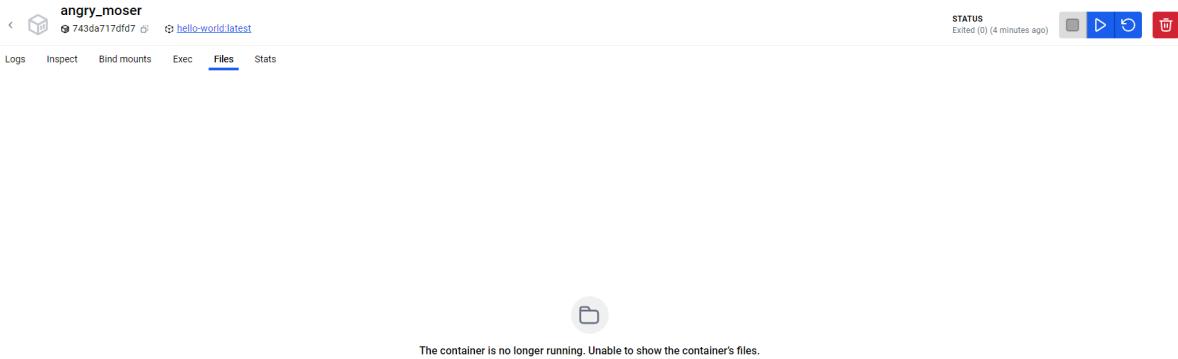
| Name | Image | Status | Port(s) |
|-------------------------------|-------------|--------|---------|
| angry_moser 743da717df07 ⚡ | hello-world | Exited | |

| Name | Image | Status | Port(s) |
|-------------------------------|-------------|--------|---------|
| angry_moser 743da717df07 ⚡ | hello-world | Exited | |

Ver Archivos de un Contenedor

- Hacemos clic en los tres puntos (:).
- Seleccionamos **View files** para explorar los archivos dentro del contenedor.

| Name | Image | Status | Port(s) |
|-------------------------------|-------------|--------|---------|
| angry_moser 743da717df07 ⚡ | hello-world | Exited | |



Gestión de imágenes Docker

Ver las Imágenes Disponibles

- En la barra lateral izquierda, seleccionamos **Images**.
- Se nos muestra la lista de imágenes descargadas en el sistema.

The screenshot shows the Docker Images page under the 'Local' tab. It lists one image: 'hello-world' (74cc54e27dc4). The table columns are Name, Tag, Status, Created, Size, and Actions. The status is 'In use' and the size is 10.07 KB. A note at the top says 'The container is no longer running. Unable to show the container's files.'

Descargar una Imagen desde Docker Hub

- Descargamos la imagen manualmente con:

```
$docker pull nginx:latest
```

- Vamos a la pestaña **Images**.
- Hacemos clic en el botón **Pull** (Descargar).
- Escribimos el nombre de la imagen y su versión, por ejemplo:

```
$nginx:latest
```

- Haz clic en **Pull** para descargar la imagen.

The screenshot shows the Docker Images page under the 'Local' tab. It lists two images: 'nginx' (97662d24417b) and 'hello-world'. The 'nginx' image has a status of 'Unused'. A context menu is open over the 'nginx' entry, with the 'Pull' option highlighted. Other options include 'View container usage', 'View packages and CVEs', and 'Push to Docker Hub'.

Eliminar una Imagen

1. Encontramos la imagen que queremos borrar en la pestaña **Images**.
2. Hacemos clic en el icono de los tres puntos (⋮) a la derecha.
3. Seleccionamos **Delete** para eliminar la imagen.

| Name | Tag | Status | Created | Size | Actions |
|-------------|----------------|--------|-------------|-----------|--|
| hello-world | 74cc54e27dc4 ⚡ | In use | 21 days ago | 10.07 KB | ⋯ Delete |
| nginx | 97662024417b ⚡ | Unused | 7 days ago | 191.99 MB | ⋯ Delete |

No se pueden eliminar imágenes que están en uso por un contenedor en ejecución:

| Name | Tag | Status | Created | Size | Actions |
|-------------|----------------|--------|-------------|----------|--|
| hello-world | 74cc54e27dc4 ⚡ | In use | 21 days ago | 10.07 KB | ⋯ Delete |

Crear una Imagen Personalizada

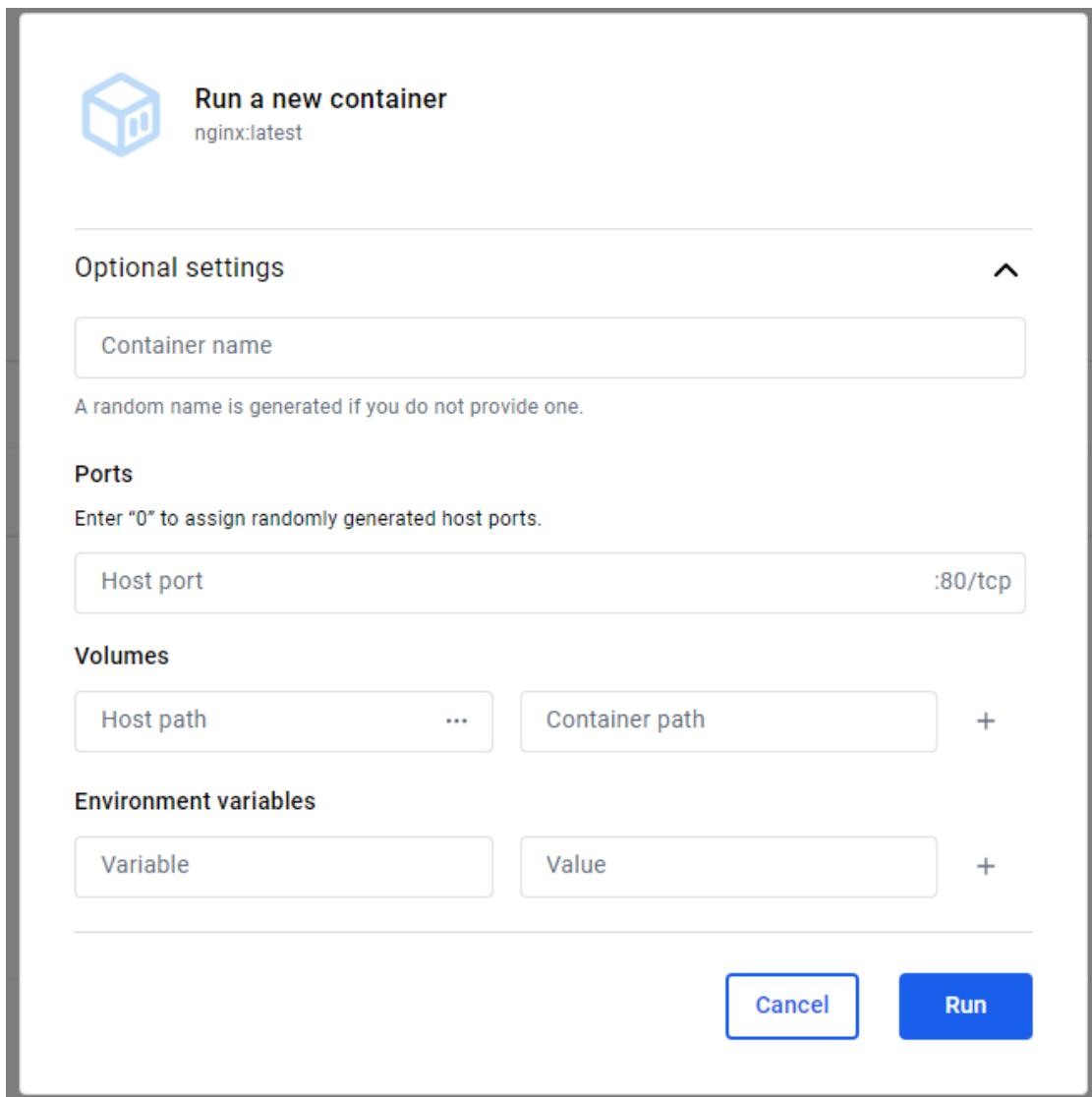
- Escribimos un **Dockerfile** con la configuración de la imagen.
- Usamos el siguiente comando en la terminal:

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

- Posteriormente, la imagen nos aparece en la pestaña **Images**.

Ejecutar un Contenedor desde una Imagen

- Encontramos la imagen que queremos ejecutar.
- Hacemos clic en **Run**.
- Configuramos los puertos y volúmenes si es necesario y hacemos clic en **Run**.



```

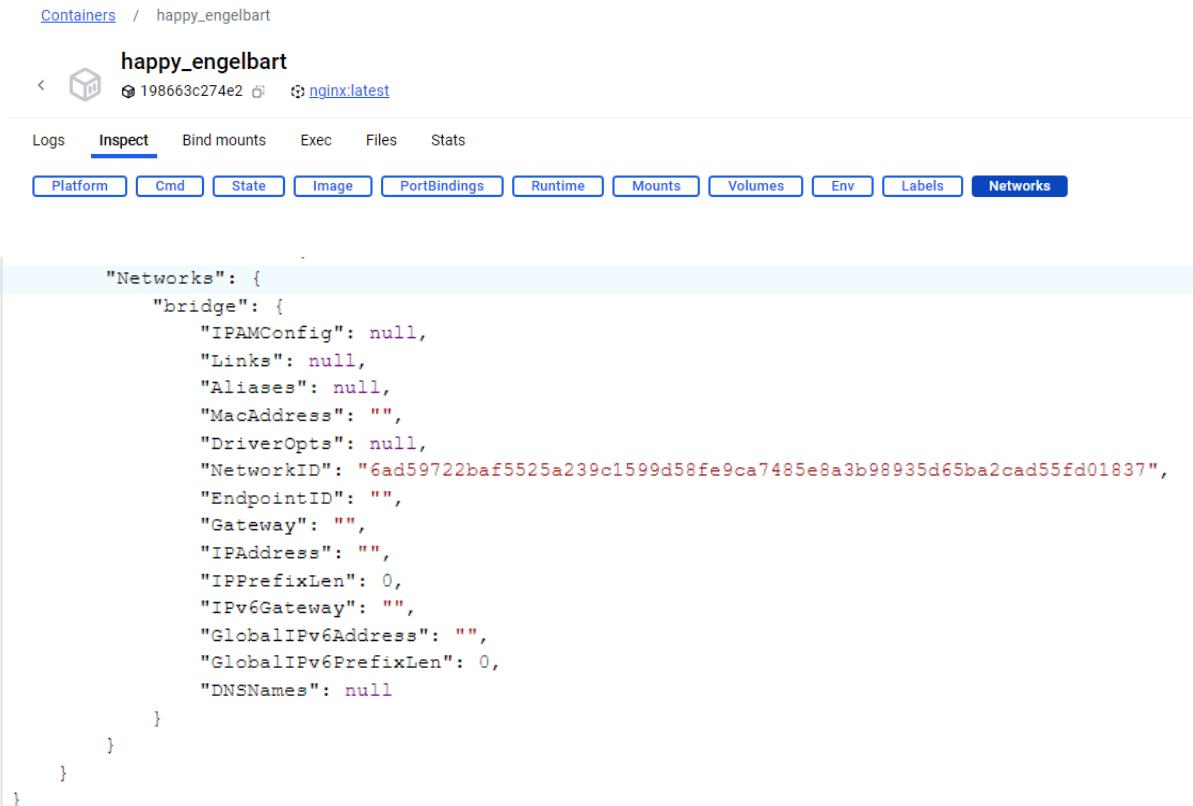
Containers / happy_engelbart
happy_engelbart 198669c274e2 ⚡ nginx:latest
STATUS Running (0 seconds ago) ✖️⟳↻✖
Logs Inspect Bind mounts Exec Files Stats
2025-02-12 11:05:37 /docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
2025-02-12 11:05:37 /docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
2025-02-12 11:05:37 /docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
2025-02-12 11:05:37 /10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
2025-02-12 11:05:37 /10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
2025-02-12 11:05:37 /docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
2025-02-12 11:05:37 /docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
2025-02-12 11:05:37 /docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
2025-02-12 11:05:37 /docker-entrypoint.sh: Configuration complete; ready for start up
2025-02-12 11:05:37 2025/02/12 10:05:37 [notice] 1#1: using the "epoll" event method
2025-02-12 11:05:37 2025/02/12 10:05:37 [notice] 1#1: nginx/1.27.4
2025-02-12 11:05:37 2025/02/12 10:05:37 [notice] 1#1: built by gcc 12.2.0-14 (Debian 12.2.0-14)
2025-02-12 11:05:37 2025/02/12 10:05:37 [notice] 1#1: OS: Linux 5.15.167.4-microsoft-standard-wSL2
2025-02-12 11:05:37 2025/02/12 10:05:37 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2025-02-12 11:05:37 2025/02/12 10:05:37 [notice] 1#1: start worker processes
2025-02-12 11:05:37 2025/02/12 10:05:37 [notice] 1#1: start worker process 29
2025-02-12 11:05:37 2025/02/12 10:05:37 [notice] 1#1: start worker process 30
2025-02-12 11:05:37 2025/02/12 10:05:37 [notice] 1#1: start worker process 31
2025-02-12 11:05:37 2025/02/12 10:05:37 [notice] 1#1: start worker process 32
2025-02-12 11:05:37 2025/02/12 10:05:37 [notice] 1#1: start worker process 33
2025-02-12 11:05:37 2025/02/12 10:05:37 [notice] 1#1: start worker process 34
2025-02-12 11:05:37 2025/02/12 10:05:37 [notice] 1#1: start worker process 35
2025-02-12 11:05:37 2025/02/12 10:05:37 [notice] 1#1: start worker process 36
2025-02-12 11:05:37 2025/02/12 10:05:37 [notice] 1#1: start worker process 37
2025-02-12 11:05:37 2025/02/12 10:05:37 [notice] 1#1: start worker process 38
2025-02-12 11:05:37 2025/02/12 10:05:37 [notice] 1#1: start worker process 39
2025-02-12 11:05:37 2025/02/12 10:05:37 [notice] 1#1: start worker process 40
2025-02-12 11:05:37 2025/02/12 10:05:37 [notice] 1#1: start worker process 41
2025-02-12 11:05:37 2025/02/12 10:05:37 [notice] 1#1: start worker process 42
2025-02-12 11:05:37 2025/02/12 10:05:37 [notice] 1#1: start worker process 43
2025-02-12 11:05:37 2025/02/12 10:05:37 [notice] 1#1: start worker process 44

```

💡 Configuración de redes y volúmenes

Ver las redes disponibles

- En la barra lateral, seleccionamos **Containers**.
- Hacemos clic en un contenedor existente y seleccionamos **View details**.
- En la pestaña **Network**, vemos las redes asociadas al contenedor.



```

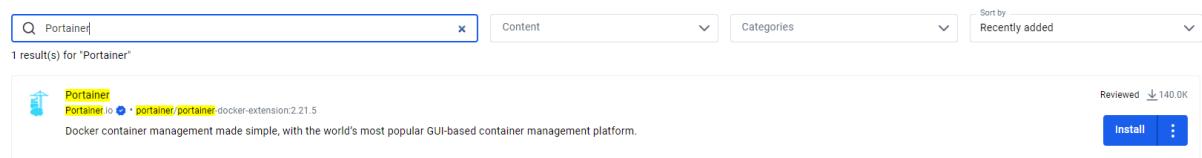
"Networks": {
    "bridge": {
        "IPAMConfig": null,
        "Links": null,
        "Aliases": null,
        "MacAddress": "",
        "DriverOpts": null,
        "NetworkID": "6ad59722baf5525a239c1599d58fe9ca7485e8a3b98935d65ba2cad55fd01837",
        "EndpointID": "",
        "Gateway": "",
        "IPAddress": "",
        "IPPrefixLen": 0,
        "IPv6Gateway": "",
        "GlobalIPv6Address": "",
        "GlobalIPv6PrefixLen": 0,
        "DNSNames": null
    }
}
}
}

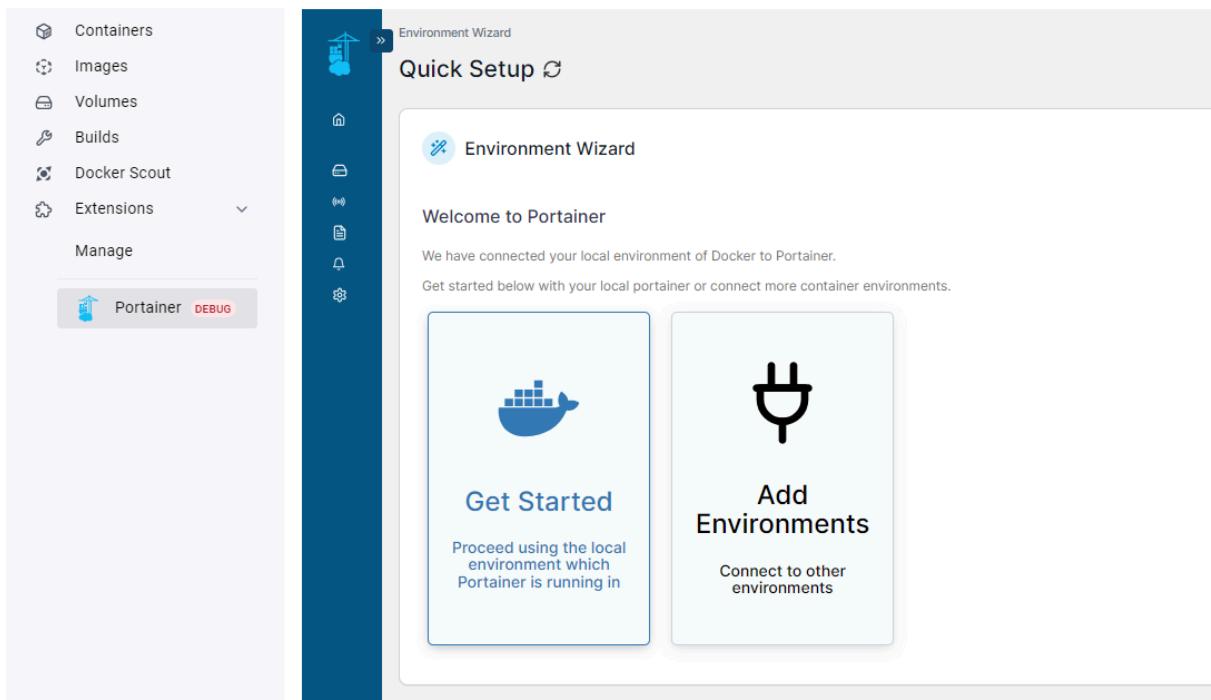
```

Crear una Nueva Red

Docker Desktop no ofrece una opción nativa para añadir redes directamente desde su interfaz, pero podemos utilizar extensiones de terceros para ampliar su funcionalidad.

- Hacemos clic en la pestaña **Extensions** en la barra lateral izquierda.
- En el Marketplace de Extensiones, buscamos **Portainer**.
- Hacemos clic en **Install** para añadir la extensión.





- Configuramos el entorno

Al iniciar Portainer por primera vez, verás la pantalla de "Quick Setup" (como en tu imagen). Aquí puedes:

- **"Get Started"**: Gestionamos el entorno local de Docker donde se ejecuta Portainer.

| Name | Type | URL | Group Name | Actions |
|-------|--------|-----------------------------|------------|-------------------------------|
| local | Docker | unix:///var/run/docker.sock | Unassigned | Manage access |

- **"Add Environments"**: Agregamos otros entornos remotos, como clústeres de Kubernetes o instancias Docker en servidores.

| Name | Type | URL | Group Name | Actions |
|-------|--------|-----------------------------|------------|-------------------------------|
| local | Docker | unix:///var/run/docker.sock | Unassigned | Manage access |

- Administramos contenedores

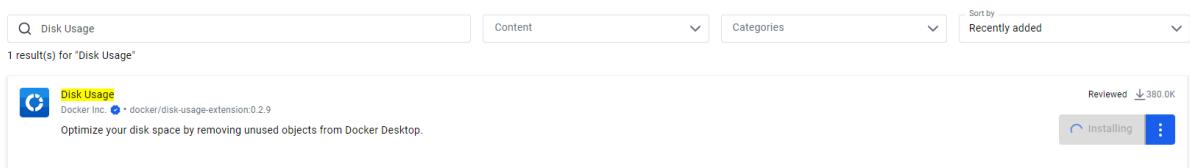
Desde el panel de control de Portainer, puedes:

- Crear, iniciar, detener y eliminar contenedores.
- Ver logs y estadísticas de rendimiento.
- Gestionar imágenes y volúmenes de Docker.
- Configurar redes y stacks con Docker Compose.

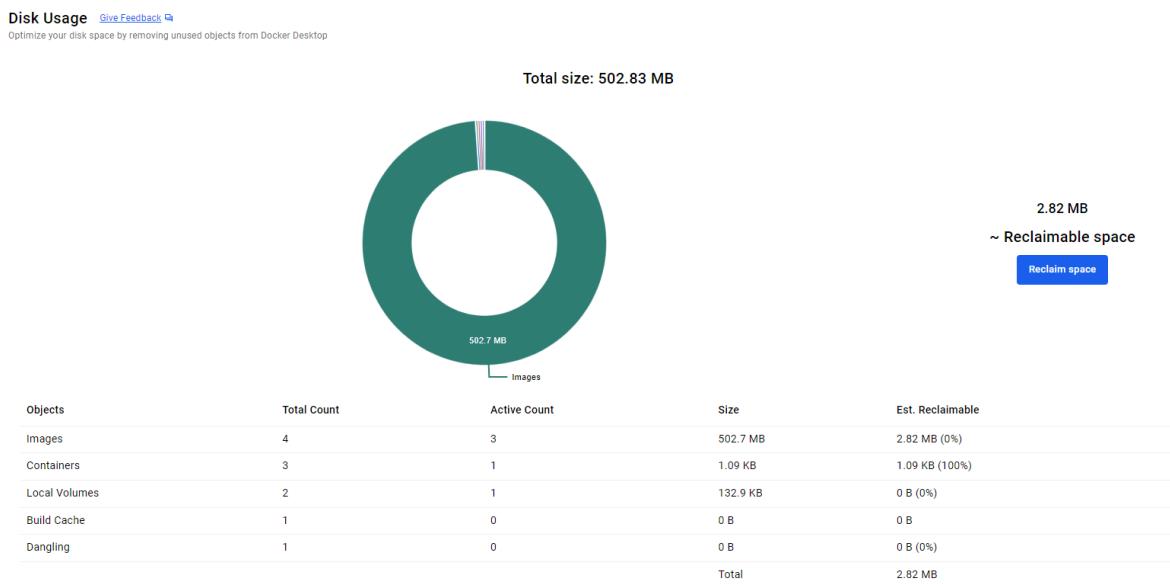
Herramientas de diagnóstico

Disk Usage (Uso de Disco)

- Muestra cuánto espacio ocupan las imágenes, contenedores y volúmenes.
- Ideal si Docker usa demasiado espacio en disco.

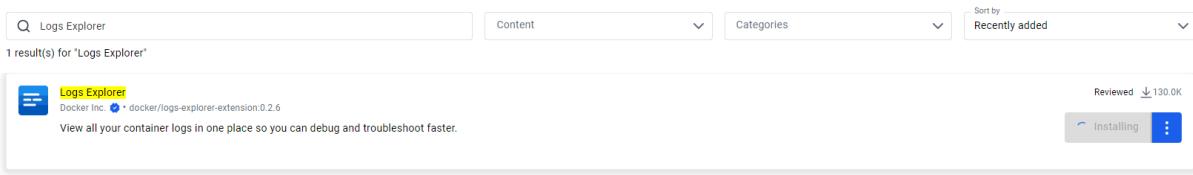


- Una vez instalada, vamos a "Extensions" > "Disk Usage".
- Revisamos el uso de espacio y liberamos lo que no necesitemos.



Logs Explorer (Explorador de Logs)

- Permite ver los logs de todos los contenedores sin usar la terminal.
- Muy útil para detectar errores sin ejecutar `docker logs`.



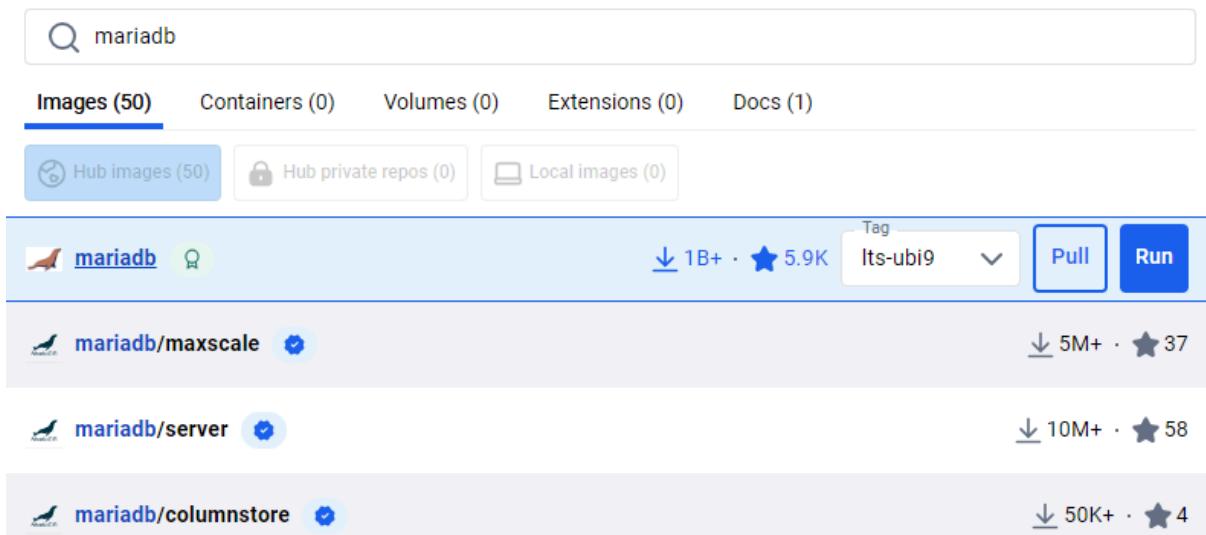
- Vamos a "**Extensions**" > "**Logs Explorer**".
- Seleccionamos el contenedor que queremos analizar y revisamos los logs en la interfaz.

| Timestamp | Container | Message |
|-----------------------|--------------------------|--|
| ! 2025-02-12 11:03:31 | [portainer_portainer...] | [90m2025/02/12 11:03AM [0m [32mINF [0m [1mgithub.com/portainer/portainer/api/http/server.go:3... |
| ! 2025-02-12 11:05:10 | [portainer_portainer...] | [90m2025/02/12 11:05AM [0m [32mINF [0m [1mgithub.com/portainer/portainer/api/jwt/jwt.go:174 [... |
| ! 2025-02-12 11:05:10 | [portainer_portainer...] | [90m2025/02/12 11:05AM [0m [32mINF [0m [1mgithub.com/portainer/portainer/api/jwt/jwt.go:174 [... |
| ! 2025-02-12 11:09:27 | [portainer_portainer...] | [90m2025/02/12 11:09AM [0m [32mINF [0m [1mgithub.com/portainer/portainer/api/jwt/jwt.go:174 [... |
| ! 2025-02-12 11:10:00 | [portainer_portainer...] | [90m2025/02/12 11:10AM [0m [32mINF [0m [1mgithub.com/portainer/portainer/api/jwt/jwt.go:174 [... |
| ! 2025-02-12 11:10:07 | [portainer_portainer...] | [90m2025/02/12 11:10AM [0m [32mINF [0m [1mgithub.com/portainer/portainer/api/jwt/jwt.go:174 [... |
| ! 2025-02-12 11:10:09 | [portainer_portainer...] | [90m2025/02/12 11:10AM [0m [32mINF [0m [1mgithub.com/portainer/portainer/api/jwt/jwt.go:174 [... |
| ! 2025-02-12 11:10:16 | [portainer_portainer...] | [90m2025/02/12 11:10AM [0m [32mINF [0m [1mgithub.com/portainer/portainer/api/jwt/jwt.go:174 [... |
| ! 2025-02-12 11:10:18 | [portainer_portainer...] | [90m2025/02/12 11:10AM [0m [32mINF [0m [1mgithub.com/portainer/portainer/api/jwt/jwt.go:174 [... |
| ! 2025-02-12 11:10:30 | [portainer_portainer...] | [90m2025/02/12 11:10AM [0m [32mINF [0m [1mgithub.com/portainer/portainer/api/jwt/jwt.go:174 [... |
| ! 2025-02-12 11:10:30 | [portainer_portainer...] | [90m2025/02/12 11:10AM [0m [32mINF [0m [1mgithub.com/portainer/portainer/api/jwt/jwt.go:174 [... |
| ! 2025-02-12 11:10:46 | [portainer_portainer...] | [90m2025/02/12 11:10AM [0m [32mINF [0m [1mgithub.com/portainer/portainer/api/jwt/jwt.go:174 [... |
| ! 2025-02-12 11:24:53 | [portainer_portainer...] | [90m2025/02/12 11:24AM [0m [32mINF [0m [1mgithub.com/portainer/portainer/api/jwt/jwt.go:174 [... |
| ! 2025-02-12 11:24:54 | [portainer_portainer...] | [90m2025/02/12 11:24AM [0m [32mINF [0m [1mgithub.com/portainer/portainer/api/jwt/jwt.go:174 [... |
| ! 2025-02-12 11:25:01 | [portainer_portainer...] | [90m2025/02/12 11:25AM [0m [32mINF [0m [1mgithub.com/portainer/portainer/api/jwt/jwt.go:174 [... |
| ! 2025-02-12 11:25:02 | [portainer_portainer...] | [90m2025/02/12 11:25AM [0m [32mINF [0m [1mgithub.com/portainer/portainer/api/jwt/jwt.go:174 [... |
| ! 2025-02-12 11:25:03 | [portainer_portainer...] | [90m2025/02/12 11:25AM [0m [32mINF [0m [1mgithub.com/portainer/portainer/api/jwt/jwt.go:174 [... |
| ! 2025-02-12 11:25:05 | [portainer_portainer...] | [90m2025/02/12 11:25AM [0m [32mINF [0m [1mgithub.com/portainer/portainer/api/jwt/jwt.go:174 [... |

Ejercicio 2: Servidor de Base de Datos

Descargar la imagen de MariaDB

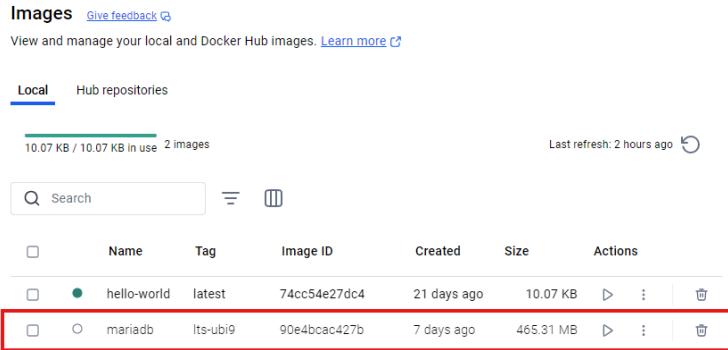
- Buscamos la imagen de **mariadb** y la descargamos
- En el menú de Docker Desktop, seleccionamos la pestaña "**Images**" (Imágenes).
- En la barra de búsqueda, escribimos "**mariadb**".
- Aparecen varias opciones relacionadas con MariaDB, seleccionamos la primera.



- Identificamos la imagen correcta con el nombre "**mariadb**" y el ícono oficial.
- Verificamos que tiene **más de 1B+ descargas y 5.9K estrellas** (esto confirma que es la imagen más utilizada y confiable).
- Hacemos clic en la imagen para seleccionarla.

Verificación que la imagen se descargó correctamente

- Vamos nuevamente a la pestaña "Images".
- Ahora, la imagen de MariaDB nos aparece en la lista de imágenes disponibles en tu sistema.



Crear un contenedor con MariaDB

- Vamos a la pestaña **Images** y haz clic en **Run** con la imagen de **MariaDB** que descargamos.
- Configuramos los siguientes parámetros en **Optional Settings**:
 - **Nombre del contenedor:** `bbdd`
 - **Puerto: 3306** (nos aseguramos de exponerlo para conexiones externas)
 - **Volumen:** Creamos un volumen llamado `datos-mariadb`.
 - En **Host path**, escribimos el nombre del volumen `datos-mariadb`.
 - En **Container path**, asignamos la ruta donde MariaDB guarda los datos dentro del contenedor `/var/lib/mysql` que es la ruta por defecto donde MariaDB almacena sus datos en el contenedor.
 - **Variables de entorno:**
 - `MYSQL_ROOT_USER=root`
 - `MYSQL_ROOT_PASSWORD=root`
 - `MYSQL_DATABASE=base`
 - `MYSQL_USER=daw`
 - `MYSQL_PASSWORD=daw`
- Guardamos y le damos a **Run**.



Run a new container

mariadb:its-ubi9

Optional settings ^

Container name

A random name is generated if you do not provide one.

Ports

Enter "0" to assign randomly generated host ports.

Host port

:3306/tcp

Volumes

Host path

...

Container path

+

Environment variables

Variable

Value

+

Cancel

Run

 Run a new container
mariadb:its-ubi9

Optional settings

Container name

A random name is generated if you do not provide one.

Ports

Enter "0" to assign randomly generated host ports.

| | | |
|-----------|------|-----------|
| Host port | 3306 | :3306/tcp |
|-----------|------|-----------|

Volumes

| | | | | | |
|-----------|---------------|-----|----------------|----------------|---|
| Host path | datos-mariadb | ... | Container path | /var/lib/mysql | + |
|-----------|---------------|-----|----------------|----------------|---|

Environment variables

| | | | | |
|----------|---------------------|-------|------|---|
| Variable | MYSQL_ROOT_USER | Value | root | - |
| Variable | MYSQL_ROOT_PASSWORD | Value | root | - |
| Variable | MARIADB_DATABASE | Value | base | - |
| Variable | MARIADB_USER | Value | daw | - |
| Variable | MARIADB_PASSWORD | Value | daw | + |

Cancel **Run**

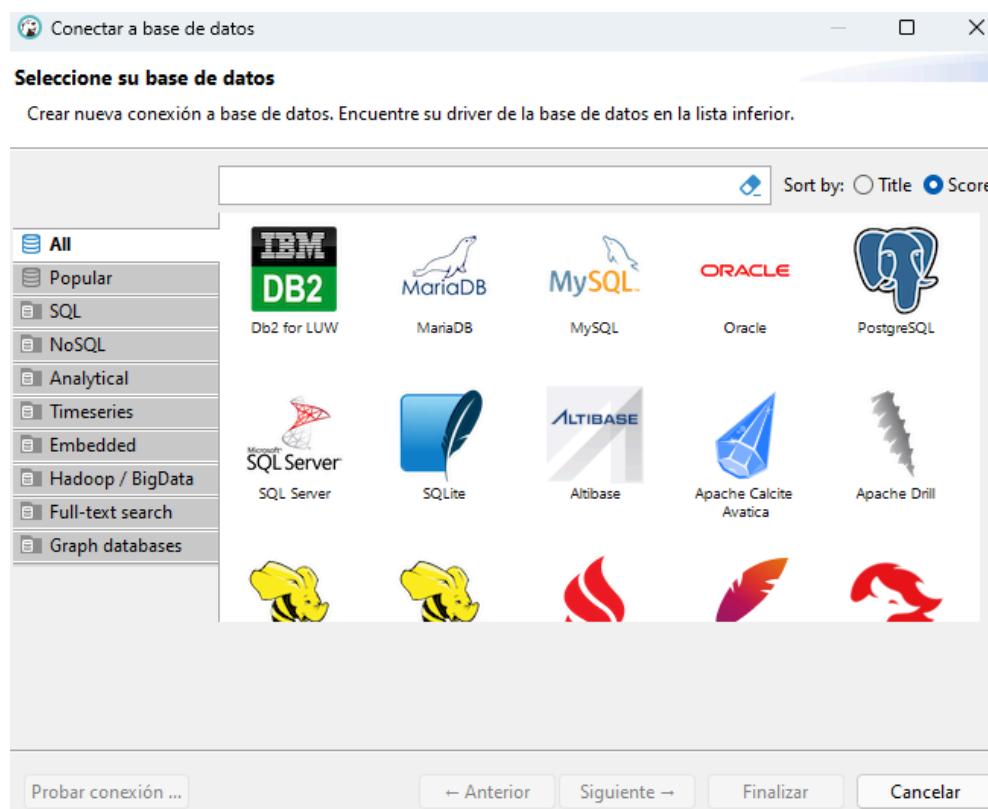
| Containers / bddd | | Logs | | Bind mounts | Exec | Files | Stats | STATUS | Running (0 seconds ago) | | | | |
|-----------------------------------|------|-----------------------------------|-------------|-------------|------|-------|-------|--------|-------------------------|--|--|--|--|
| | bddd | ⌚ 7ed85ccbb851 ⌚ mariadb:its-ubi9 | 3306:3306 C | | | | | | | | | | |
| | | | | | | | | | | | | | |

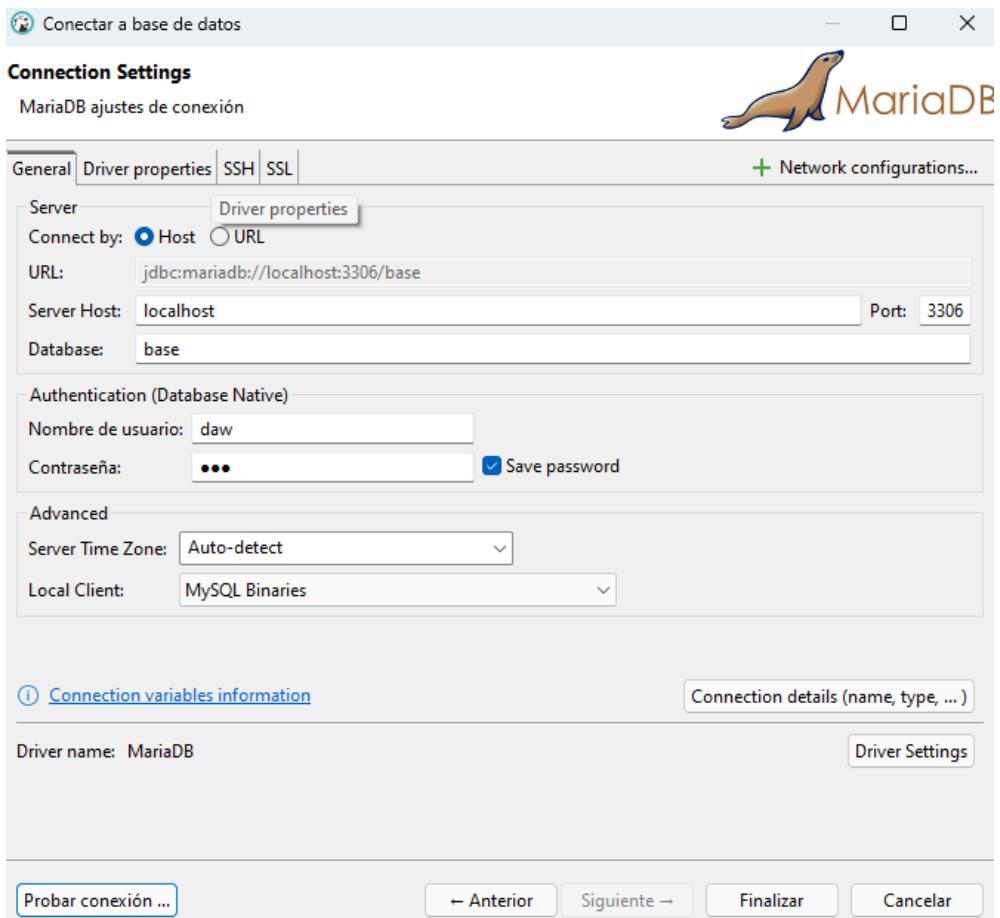
```
2025-02-12 12:06:00 2025-02-12 11:06:00+00:00 [Note] [Entrypoint]: Entrypoint script for MariaDB Server 11.4.5 started.
2025-02-12 12:06:00 2025-02-12 11:06:00+00:00 [Note] [Entrypoint]: Initializing database files
2025-02-12 12:06:01 2025-02-12 11:06:01+00:00 [Note] [Entrypoint]: Files initialized
2025-02-12 12:06:01 2025-02-12 11:06:01+00:00 [Note] [Entrypoint]: Starting temporary server
2025-02-12 12:06:02 2025-02-12 11:06:02+00:00 [Note] [Entrypoint]: Waiting for server startup
2025-02-12 12:06:02 2025-02-12 11:06:02+00:00 [Note] Starting MariaDB 11.4.5-MariaDB source revision 077110260ff5c04216af4bf1243c65f8c67ccf4 server_vld KjyTAYmzkiwRQBEIh5uvt04gpeY= as process 74
2025-02-12 12:06:02 2025-02-12 11:06:02+00:00 [Note] InnoDB: Compressed tables use zlib 1.2.11
2025-02-12 12:06:02 2025-02-12 11:06:02+00:00 [Note] InnoDB: Number of transaction pools: 1
2025-02-12 12:06:02 2025-02-12 11:06:02+00:00 [Note] InnoDB: crc32 + pcInnodb instructions
2025-02-12 12:06:02 2025-02-12 11:06:02+00:00 [Note] mariadb: _OTMPFILE is not supported on /tmp (disabling future attempts)
2025-02-12 12:06:02 2025-02-12 11:06:02+00:00 [Note] InnoDB: Using liburing
2025-02-12 12:06:02 2025-02-12 11:06:02+00:00 [Note] InnoDB: Initializing buffer pool, total size = 128.000MB, chunk size = 2.000MB
2025-02-12 12:06:02 2025-02-12 11:06:02+00:00 [Note] InnoDB: Completed initialization of buffer pool
2025-02-12 12:06:02 2025-02-12 11:06:02+00:00 [Note] InnoDB: File custom buffer for innodb_mixed /blank e100-AAA buffer
```

Conectarse a la base de datos con DBeaver

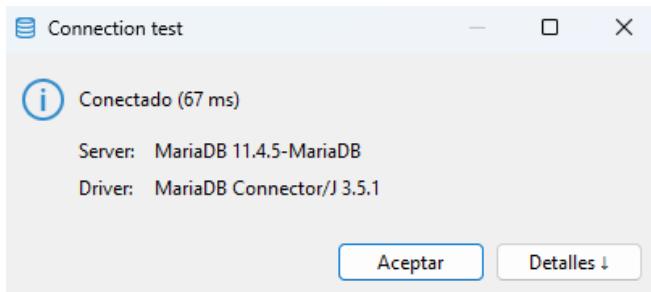
- Abrimos **DBeaver**
- Creamos una nueva conexión:
 - Configuración general:

- **Host:** localhost
- **Puerto:** 3306
- Credenciales:
 - **Usuario:** daw
 - **Contraseña:** daw
- Base de datos:
 - **Base de datos:** base





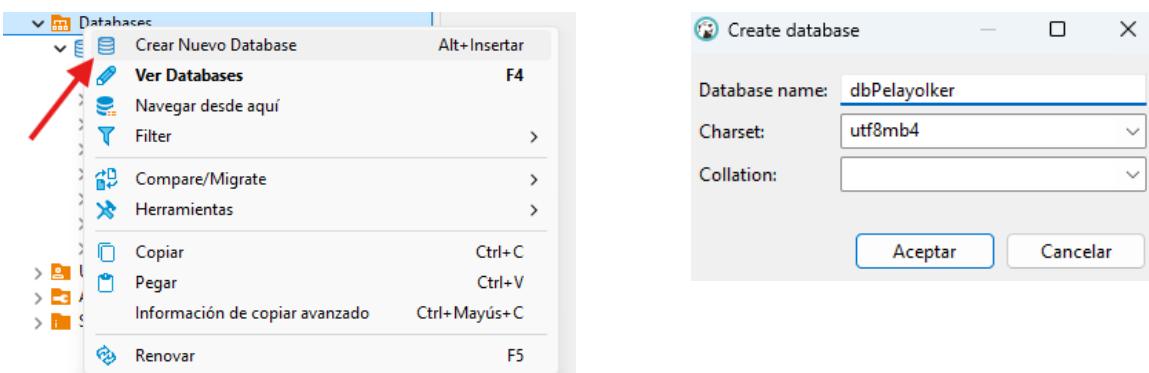
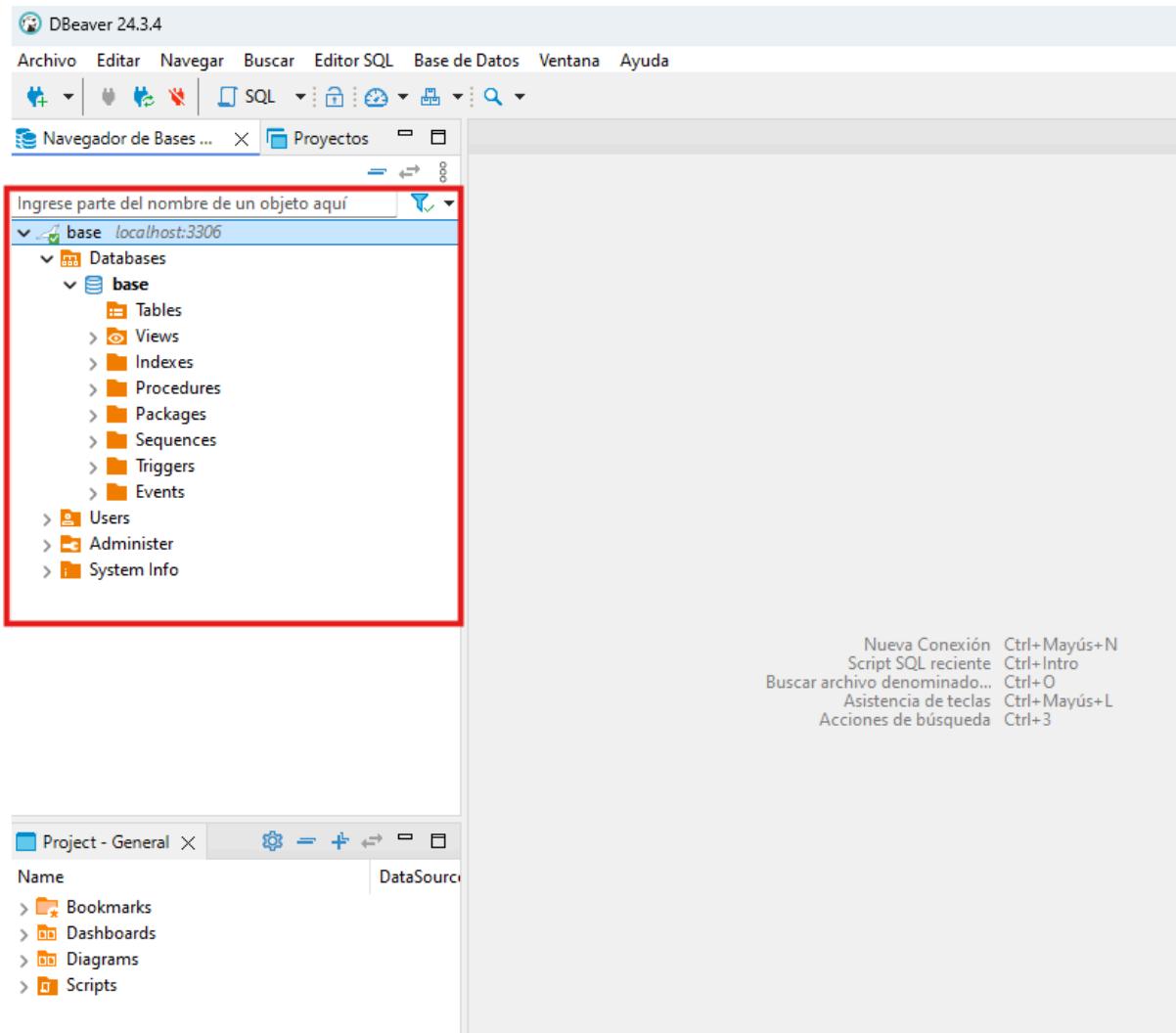
Probamos la conexión y nos aseguramos de que funciona.



Crear una nueva base de datos

- Hacemos clic derecho en la conexión con **MariaDB** y seleccionamos "**Crear Nuevo Database**".
- Se nos abre una ventana llamada "**Create database**".
- En **Database name**, escribimos el nombre de la base de datos, en este caso la llamamos **dbPelayolker**.
- En **Charset**, seleccionamos **utf8mb4** (para compatibilidad con caracteres especiales y emojis).

- Opcionalmente, podíamos configurar **Collation**, pero en este caso lo dejarlo vacío para la opción por defecto.
- Hacemos clic en "**Aceptar**" para crear la base de datos.



Seleccionamos la base de datos y crear una tabla

- Hacemos clic derecho sobre la base de datos `dbPelayolker` que acabamos de crear.
- Luego, hacemos clic derecho en "**Tables**" → "**Create New Table**".
- Asignamos un nombre a la tabla , le ponemos por ejemplo, `ejemplo` .
- Agregamos las columnas necesarias:
 - `id` → **INT**, los marcamos como **Primary Key**, opción **Auto Increment**.
 - `nombre` → **VARCHAR(50)**, lo marcamos como **NOT NULL**.
- Guardamos los cambios y presionamos en "**Aceptar**".

The screenshot shows the MySQL Workbench interface. At the top, there's a search bar with placeholder text "Ingrese parte del nombre de un objeto aquí". Below it is a tree view of the database structure under "base localhost:3306". A red box highlights the "dbPelayolker" database node. The tree also includes "Databases", "base", "sys", "Users", "Administer", and "System Info". In the bottom half, there's a script editor titled "Script-1" containing the following SQL code:

```

CREATE TABLE ejemplo (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(50) NOT NULL
);

```

Verificamos que la tabla se creo correctamente

- Expandimos la base de datos `dbPelayolker` en el panel izquierdo y hacemos clic en "**Tables**".
- Vemos la tabla `ejemplo` , con lo cual se ha creado correctamente.

Ingrese parte del nombre de un objeto aquí

base localhost:3306

- Databases
 - base
 - Tables
 - Views
 - Indexes
 - Procedures
 - Packages
 - Sequences
 - Triggers
 - Events
 - dbPelayolker
 - Tables
 - ejemplo
 - Views
 - Indexes
 - Procedures
 - Packages
 - Sequences
 - Triggers
 - Events
 - sys
- Users
- Administer
- System Info

Detener y eliminar los contenedores

- Abrimos **Docker Desktop**.
- En la pestaña **Containers**, buscamos el contenedores **bbdd**.
- Detenemos cada contenedor presionando el botón **Stop (■)**.
- Luego, eliminamos los contenedores con el botón **Delete (☒)**.

Containers [Give feedback](#)

View all your running containers and applications. [Learn more](#)

Container CPU usage [\(1\)](#) 0.01% / 1600% (16 CPUs available)

Container memory usage [\(1\)](#) 314.8MB / 15.19GB

Show charts

| <input type="checkbox"/> | Name | Container ID | Image | Port(s) | CPU (%) | Last started | Actions |
|-------------------------------------|-------------|--------------|------------------|-------------|---------|----------------|--|
| <input type="checkbox"/> | sad_galileo | 1db91595bbe9 | hello-world | | 0% | 6 days ago | ▶ ⋮ Delete |
| <input checked="" type="checkbox"/> | bbdd | 7ed85ccb851 | mariadb:its-ubi9 | 3306:3306 ↗ | 0.01% | 30 minutes ago | ▶ ⋮ Delete |

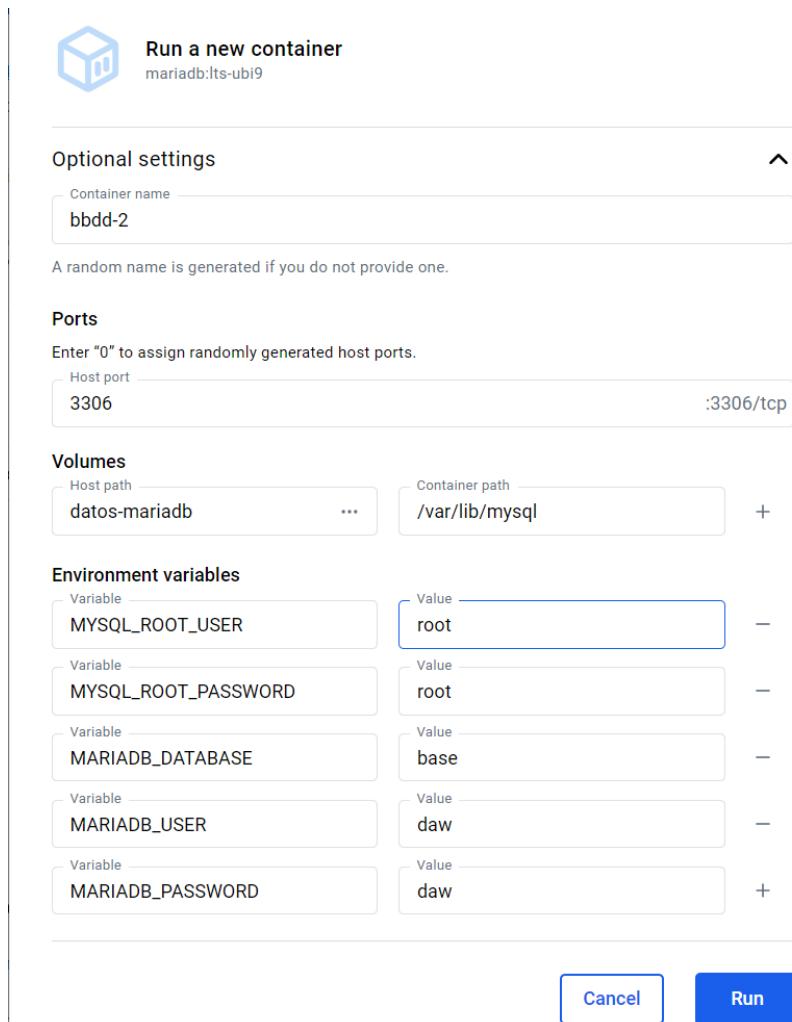
Verificamos que el volumen "datos-mariadb" sigue existiendo

- En Docker Desktop, vamos a la pestaña **Volumes**.
- Buscamos el volumen **datos-mariadb**.
- Como se ve el volumen sigue ahí, lo que significa que los datos no se han perdido.

| Volumes | | | | | | |
|---|---------------|---------|---------------|---------|---------|------|
| Manage your volumes, view usage, and inspect their contents. Learn more | | | | | | |
| Search | | Actions | | | Created | Size |
| <input type="checkbox"/> | datos-mariadb | | 4 minutes ago | 0 Bytes | | |

Creamos otro contenedor con el mismo volumen

- En Docker Desktop, vamos a **Images** y buscamos **MariaDB**.
- Hacemos clic en **Run** para desplegar un nuevo contenedor.
- Configuramos:
 - **Container name:** `bbdd-2`
 - **Port:** `3306`
 - **Volume:** `datos-mariadb`
 - **Environment variables** (como antes):
 - `MYSQL_ROOT_USER=root`
 - `MYSQL_ROOT_PASSWORD=root`
 - `MARIADB_DATABASE=base`
 - `MARIADB_USER=daw`
 - `MARIADB_PASSWORD=daw`
- Hacemos clic en **Run**.



Comprobamos que los datos siguen en el nuevo contenedor

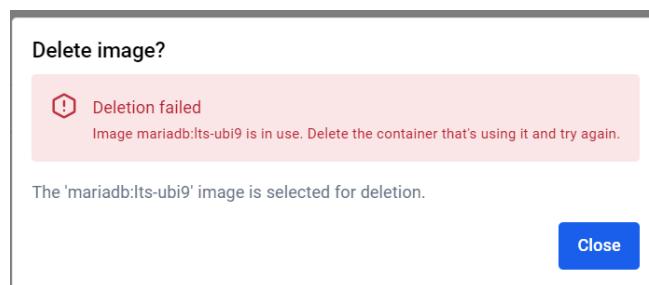
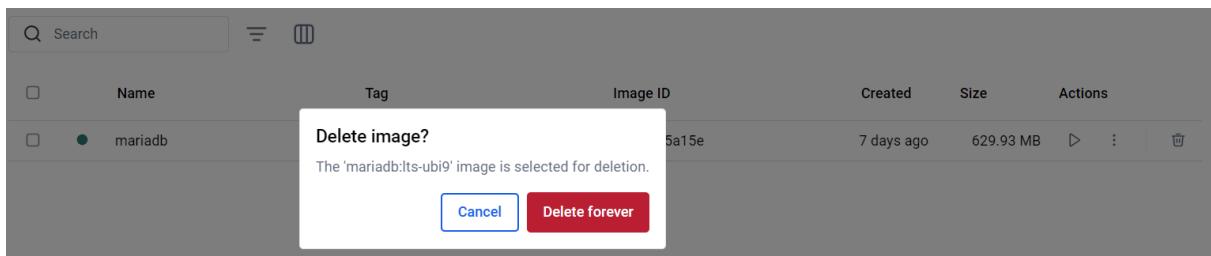
- Abrimos **DBeaver** y nos **conectatamos de nuevo a MariaDB** (`localhost:3306` , usuario `daw`).
- Verificamos que `dbPelayolker` y la tabla `ejemplo` siguen existiendo.
- Como todo está correcto, los datos se han conservado gracias al volumen persistente.



Intentar borrar la imagen de MariaDB

- En Docker Desktop, vamos a la pestaña **Images**.

- Intentamos eliminar la imagen de **MariaDB** ([Delete](#)).
- ¿Qué sucede?
 - No podremos eliminar si hay contenedores en ejecución usándola ([bdd-2](#)).
 - Tenemos que eliminar [bdd-2](#) antes de poder borrar la imagen.



Borrar todo (Volumen, Imagen y Contenedor)

- Eliminamos los contenedores [bdd](#) y [bdd-2](#) desde Docker Desktop.

| Containers | | | | | | | |
|--------------------------|-------|--------------|---------------------------------|-----------|---------|----------------|---|
| <input type="checkbox"/> | Name | Container ID | Image | Port(s) | CPU (%) | Last started | Actions |
| <input type="radio"/> | bdd | 94d4bd08116c | mariadb:ts-ubi9 | 3306:3306 | N/A | 1 hour ago | View ⋮ Delete |
| <input type="radio"/> | bdd-2 | ff1bf68bdf5d | mariadb:ts-ubi9 | 3306:3306 | N/A | 16 minutes ago | View ⋮ Delete |

- Eliminamos el volumen [datos-mariadb](#) desde la pestaña **Volumes**.

| Name | Created | Size | Actions |
|---------------|----------------|---------|---|
| datos-mariadb | 17 minutes ago | 0 Bytes | View ⋮ Delete |

- Eliminamos la imagen de MariaDB desde la pestaña **Images**.

| Images | | | | | | | |
|--------------------------|---------|---------|--------------|------------|-----------|---|------------------------|
| <input type="checkbox"/> | Name | Tag | Image ID | Created | Size | Actions | Delete |
| <input type="radio"/> | mariadb | ts-ubi9 | 05f189b5a15e | 7 days ago | 629.93 MB | View ⋮ Delete | Delete |

Ejercicio 3: Contenedores en Red

Crear una red bridge (reldb)

En **Docker Desktop**, no hay una sección visual para redes en la interfaz, pero podemos verlas y administrarlas con comandos Docker.

Primero listamos las redes, luego, creamos la red bridge pedida y por ultimo vemos los detalles de esa red y comprobamos que se ha creado correctamente:

```
$ docker network ls  
$ docker network create reldb  
$ docker network inspect reldb
```

```
C:\Windows\System32>docker network ls  
NETWORK ID      NAME      DRIVER      SCOPE  
a442ae9f5d9b    bridge    bridge      local  
5ad4578655cc    host      host       local  
f0d37d297057    none      null       local  
  
C:\Windows\System32>docker network create reldb  
9dbf0118bc4ca006697b866f3f87478e13ae0c36bf533f85b52e9fe3ea7d8834
```

```
C:\Windows\System32>docker network inspect redbd
[
    {
        "Name": "redbd",
        "Id": "9dbf0118bc4ca006697b866f3f87478e13ae0c36bf533f85b52e9fe3ea7d8834",
        "Created": "2025-02-12T18:19:37.719807195Z",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": {},
            "Config": [
                {
                    "Subnet": "172.18.0.0/16",
                    "Gateway": "172.18.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {},
        "Options": {},
        "Labels": {}
    }
]
```



Crear el contenedor de MariaDB con comandos

Ejecutamos en la terminal:

```
$ docker run -d --name mariadb-container \
--network redbd \
-e MYSQL_ROOT_PASSWORD=root \
-e MYSQL_DATABASE=base \
-e MYSQL_USER=daw \
-e MYSQL_PASSWORD=daw \
-p 3306:3306 \
-v mariadb_data:/var/lib/mysql \
mariadb:latest
```

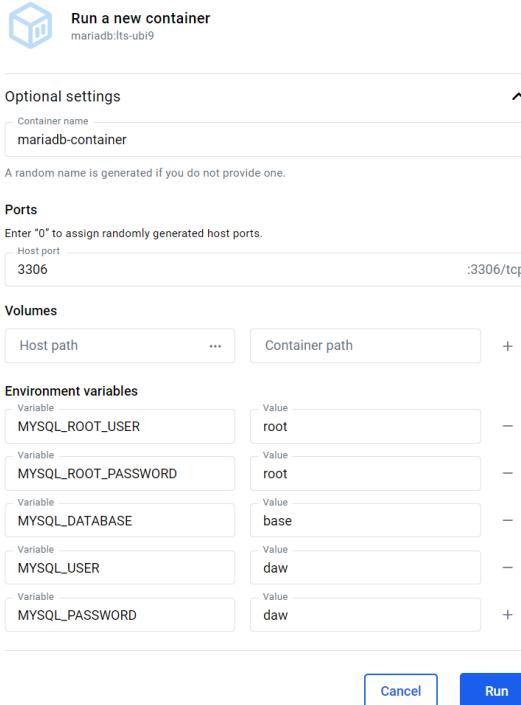
```

PS C:\Windows\system32> docker run -d --name mariadb-container
>> --network redbd
>> -e MYSQL_ROOT_PASSWORD=root
>> -e MYSQL_DATABASE=base
>> -e MYSQL_USER=daw
>> -e MYSQL_PASSWORD=daw
>> -p 3306:3306
>> -v mariadb_data:/var/lib/mysql
>> mariadb:latest
>>

Unable to find image 'mariadb:latest' locally
latest: Pulling from library/mariadb
2cbd49ab14b1: Download complete
f67c6fb0ef5: Download complete
640331c2cc76: Download complete
1f731489858b: Download complete
65dd09f27c61: Download complete
edb426f4a1af: Download complete
760f6e3db6bf: Download complete
5a7813e071bf: Download complete
Digest: sha256:bfb1298c06cd15f446f1c59600b3a856dae861705d1a2bd2a00edb6c74ba748
Status: Downloaded newer image for mariadb:latest
e8c91772fc2d208558f8bcb5cc6eff647c6758b2192a35dc37f4aa588820390a

```

Crear el contenedor de MariaDB con Docker Desktop



Run a new container
mariadb:bits-ubi9

Optional settings

Container name: mariadb-container

A random name is generated if you do not provide one.

Ports

Enter "0" to assign randomly generated host ports.

Host port: 3306 Container port: 3306/tcp

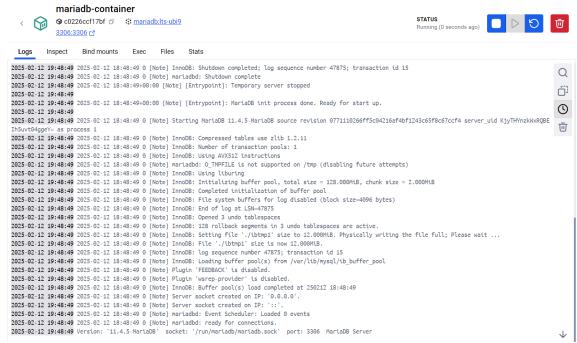
Volumes

| | | | |
|-----------|-----|----------------|---|
| Host path | ... | Container path | + |
|-----------|-----|----------------|---|

Environment variables

| | | |
|-------------------------------|-------------|---|
| Variable: MYSQL_ROOT_USER | Value: root | - |
| Variable: MYSQL_ROOT_PASSWORD | Value: root | - |
| Variable: MYSQL_DATABASE | Value: base | - |
| Variable: MYSQL_USER | Value: daw | - |
| Variable: MYSQL_PASSWORD | Value: daw | + |

Cancel Run



Podemos ver en los logs que el contenedor de MariaDB se ha iniciado correctamente. El sistema nos muestra una secuencia de mensajes de inicialización que confirman que la base de datos está lista para aceptar conexiones y que todos los componentes necesarios se han cargado correctamente.

Container CPU usage ⓘ
0.02% / 800% (8 CPUs available)

Container memory usage ⓘ
295.9MB / 7.43GB

Show charts

| | Name | Container ID | Image | Port(s) | CPU (%) | Last started | Actions |
|--------------------------|-------------------|--------------|-----------------|-------------|---------|----------------|---|
| <input type="checkbox"/> | mariadb-container | c0226ccf17bf | mariadb:ts-ubi9 | 3306:3306 ⚡ | 0.02% | 47 seconds ago | <input type="checkbox"/> ⋮ <input type="checkbox"/> |

Volumes [Give feedback ⓘ](#)
Manage your volumes, view usage, and inspect their contents. [Learn more ⓘ](#)

| | Name ↑ | Created | Size | Actions |
|--------------------------|---|--------------|---------|---|
| <input type="checkbox"/> | c98d4db027a314cd902d9e27d15f0b9b51e092bca31887b1fc1ee12be46a8 | 1 minute ago | 0 Bytes | <input type="checkbox"/> ⋮ <input type="checkbox"/> |

🌐 Crear el contenedor con Adminer o phpMyAdmin con comandos

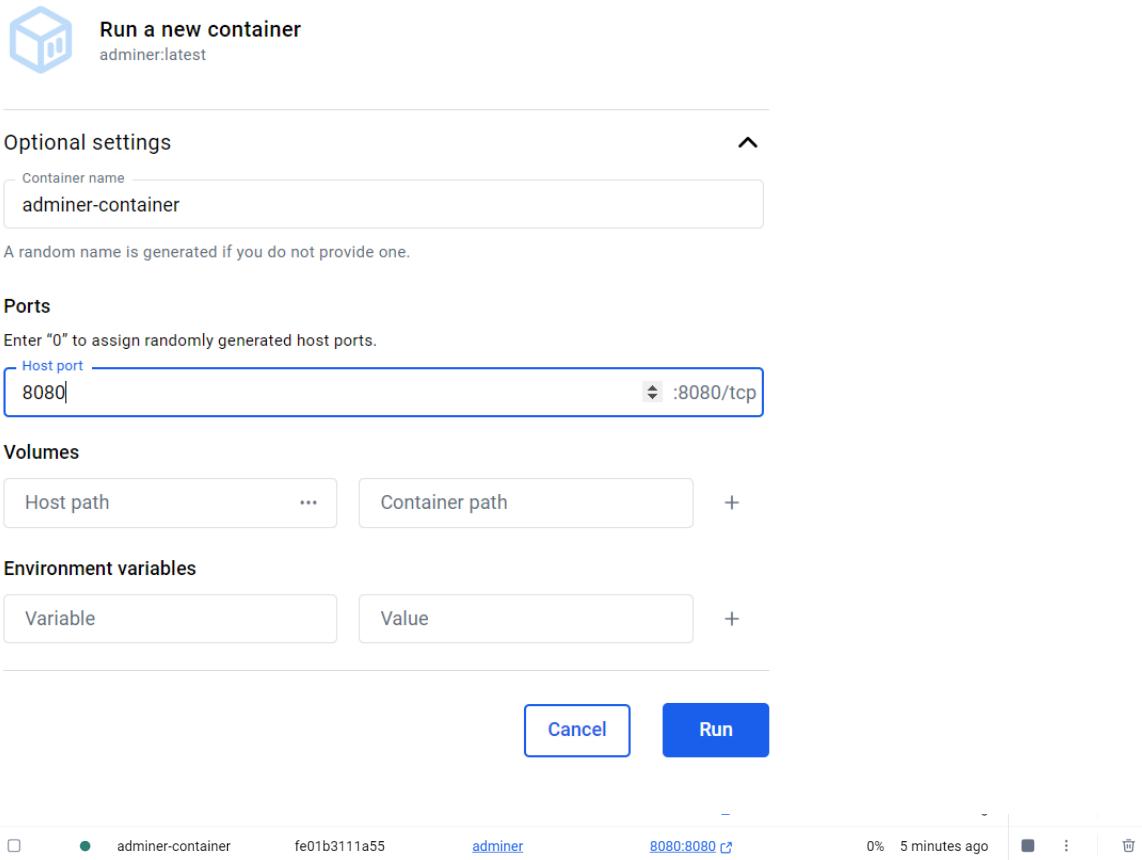
```
$docker run -d --name adminer-container \
--network redbd \
-p 8080:8080 \
adminer
```

```
PS C:\Windows\system32> docker run -d --name adminer-container \
>>   --network redbd \
>>   -p 8080:8080 \
>>   adminer
>>
Unable to find image 'adminer:latest' locally
latest: Pulling from library/adminer
c82cd9b427d9: Download complete
ed94e1c95a57: Download complete
9a4cd7b75371: Download complete
798a45c9628c: Download complete
884bce373183: Download complete
73226dab8db5: Download complete
574dfab7cda2: Download complete
Digest: sha256:34d37131366c5aa84e1693dbed48593ed6f95fb450b576c1a7a59d3a9c9e8802
Status: Downloaded newer image for adminer:latest
fe01b3111a553aac63aac55e40ee1590564ea31f511daef9e2eadc605aeb6772
```

🌐 Crear el contenedor con Adminer o phpMyAdmin con Docker Desktop

- Vamos a la pestaña "**Images**" y hacemos clic en "**Run**" con la imagen de Adminer.
- En "**Advanced settings**":
 - Nombre: **adminer-container**.

- Host Port: **8080**.
- Hacemos clic en "Run".



▀ Acceder a la base de datos desde la interfaz gráfica

Abrimos el navegador y vamos a <http://localhost:8080>.

The screenshot shows a web browser displaying the Adminer 4.8.1 login page. The language is set to Spanish. The login form contains the following fields:

| | |
|-------------------------------|-------|
| Motor de base de datos | MySQL |
| Servidor | db |
| Usuario | |
| Contraseña | |
| Base de datos | |

At the bottom of the form are two buttons: **Login** and **Guardar contraseña**.

- Rellenamos los datos de conexión:
 - **Servidor:** `mariadb-container` (nombre del contenedor).
 - **Usuario:** `daw`.
 - **Contraseña:** `daw`.
 - **Base de datos:** `base`.
- Hacemos clic en "Login".

Idioma: `Español`

Adminer 4.8.1

Login

| | |
|-------------------------------|--------------------------------|
| Motor de base de datos | MySQL |
| Servidor | <code>mariadb-container</code> |
| Usuario | <code>daw</code> |
| Contraseña | <code>...</code> |
| Base de datos | <code>base</code> |

Guardar contraseña

← → ⌂ ⌂ localhost:8080/?server=mariadb-container&username=daw&db=base

Idioma: `Español`

Adminer 4.8.1

DB: `base`

[Comando SQL](#) [Importar](#)
[Exportar](#) [Crear tabla](#)

No existen tablas.

MySQL » `mariadb-container` » Base de datos: `base`

Base de datos: `base`

[Modificar Base de datos](#) [Esquema de base de datos](#) [Privilegios](#)

Tablas y vistas

No existen tablas.

[Crear tabla](#) [Crear vista](#)

Procedimientos

[Crear procedimiento](#) [Crear función](#)

Eventos

[Crear Evento](#)

Crear una Tabla en la Base de Datos

- Entramos en la base de datos `base` y elegimos "**Crear tabla**".

- Definimos una tabla de ejemplo:

| Campo | Tipo | Clave primaria |
|--------|-------------|--|
| id | INT | <input checked="" type="checkbox"/> (Auto-incremental) |
| nombre | VARCHAR(50) | <input type="checkbox"/> |

- Guardamos los cambios.

Idioma: Español

MySQL » mariadb-container » base » Crear tabla

Adminer 4.8.1

Crear tabla

Nombre de la tabla: ejemplo (motor) (colación) Guardar

| Nombre de columna | Tipo | Longitud | Opciones | NULL | AI? | + |
|-------------------|---------|----------|------------|--------------------------|----------------------------------|---|
| id | int | | | <input type="checkbox"/> | <input checked="" type="radio"/> | <input type="button" value="+"/> <input type="button" value="↑"/> <input type="button" value="↓"/> <input type="button" value="x"/> |
| nombre | varchar | 50 | (colación) | <input type="checkbox"/> | <input type="radio"/> | <input type="button" value="+"/> <input type="button" value="↑"/> <input type="button" value="↓"/> <input type="button" value="x"/> |

Incremento automático: Valores predeterminados Comentario

Idioma: Español

MySQL » mariadb-container » base » Tabla: ejemplo

Adminer 4.8.1

Tabla: ejemplo

Tabla creada. 19:28:31 Comando SQL

Visualizar contenido Mostrar estructura Modificar tabla Nuevo Registro

| Columna | Tipo | Comentario |
|---------|-------------------------------|------------|
| id | int(11) Incremento automático | |
| nombre | varchar(50) | |

Índices

Claves externas

Disparadores

✍ Borrar los contenedores, la red y los volúmenes utilizados con comandos

```
$docker stop mariadb-container adminer
$docker rm mariadb-container adminer
$docker network rm my_network
$docker volume prune -f
```

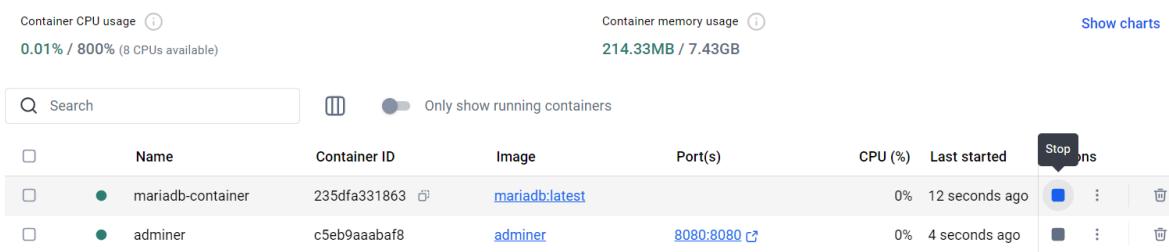
```

PS C:\Windows\system32> docker stop mariadb-container adminer
mariadb-container
adminer
PS C:\Windows\system32> docker rm mariadb-container adminer
mariadb-container
adminer
PS C:\Windows\system32> docker network rm my_network
my_network
PS C:\Windows\system32> docker volume prune -f
Deleted Volumes:
8e9a6bd9818e247f1aaaa94df39354bdd728d696cf7f40bad37ec1a21c19276e
Total reclaimed space: 156MB

```

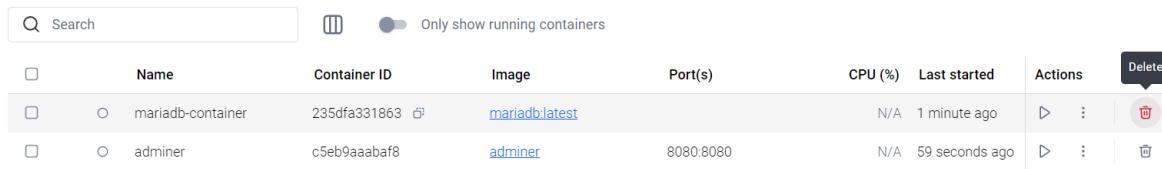
Borrar los contenedores, la red y los volúmenes utilizados con Docker Desktop

Primero paramos los contenedores, haciendo click en Stop.



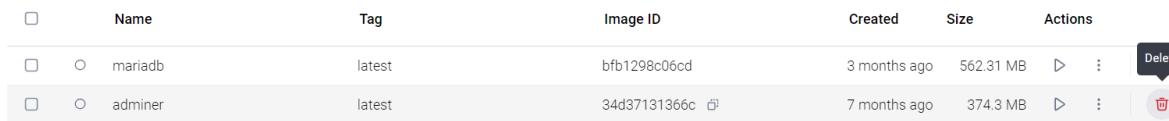
| | Name | Container ID | Image | Port(s) | CPU (%) | Last started | Actions |
|--------------------------|-------------------|--------------|--------------------------------|-----------|---------|----------------|---|
| <input type="checkbox"/> | mariadb-container | 235dfa331863 | mariadb:latest | | 0% | 12 seconds ago | Stop ⋮ trash |
| <input type="checkbox"/> | adminer | c5eb9aaabaf8 | adminer | 8080:8080 | 0% | 4 seconds ago | Stop ⋮ trash |

Y luego los borramos:



| | Name | Container ID | Image | Port(s) | CPU (%) | Last started | Actions |
|--------------------------|-------------------|--------------|--------------------------------|-----------|---------|----------------|---|
| <input type="checkbox"/> | mariadb-container | 235dfa331863 | mariadb:latest | | N/A | 1 minute ago | Delete ⋮ trash |
| <input type="checkbox"/> | adminer | c5eb9aaabaf8 | adminer | 8080:8080 | N/A | 59 seconds ago | Delete ⋮ trash |

Una vez borrados podemos borrar sus imágenes:



| | Name | Tag | Image ID | Created | Size | Actions |
|--------------------------|---------|--------|--------------|--------------|-----------|---|
| <input type="checkbox"/> | mariadb | latest | bfb1298c06cd | 3 months ago | 562.31 MB | Delete ⋮ trash |
| <input type="checkbox"/> | adminer | latest | 34d37131366c | 7 months ago | 374.3 MB | Delete ⋮ trash |

Ejercicio 4: Docker Compose

Crear el archivo `docker-compose.yaml` con comandos

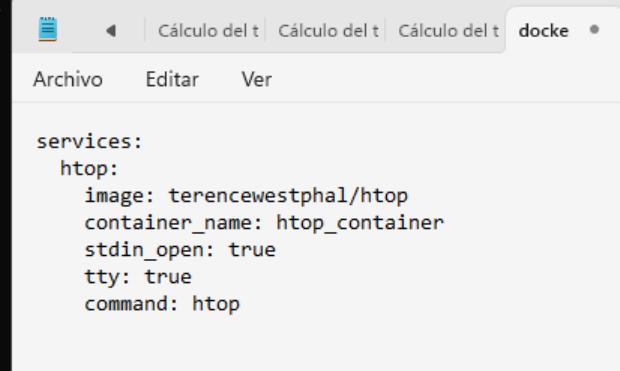
Abrimos una terminal y creamos una nueva carpeta para el proyecto, dentro de esta carpeta, creamos el archivo `docker-compose.yaml`. Para esto, usamos bloc de notas(o cualquier otro editor de texto):

```
$mkdir htop-docker && cd htop-docker  
$notepad docker-compose.yaml
```

```
C:\Windows\System32\htop-docker>docker version  
Client:  
  Version:          27.4.0  
  API version:     1.47  
  Go version:      go1.22.10  
  Git commit:      bde2b89  
  Built:           Sat Dec  7 10:40:21 2024  
  OS/Arch:         windows/amd64  
  Context:         desktop-linux  
  
Server: Docker Desktop 4.37.1 (178610)  
Engine:  
  Version:          27.4.0  
  API version:     1.47 (minimum version 1.24)  
  Go version:      go1.22.10  
  Git commit:      92a8393  
  Built:           Sat Dec  7 10:38:57 2024  
  OS/Arch:         linux/amd64  
  Experimental:    false  
containerd:  
  Version:          1.7.21  
  GitCommit:        472731909fa34bd7bc9c087e4c27943f9835f111  
runc:  
  Version:          1.1.13  
  GitCommit:        v1.1.13-0-g58aa920  
docker-init:  
  Version:          0.19.0  
  GitCommit:        de40ad0
```

```
C:\Windows\System32\htop-docker>notepad docker-compose.yaml
```

```
C:\Windows\System32\htop-docker>
```



Pegamos el siguiente contenido en el archivo y luego lo guardamos y cerramos el bloc de notas:

```
services:  
  htop:  
    image: terencewestphal/htop  
    container_name: htop_container  
    stdin_open: true
```

```
tty: true  
command: htop
```

Crear el archivo `docker-compose.yaml` sin comandos

- Abrimos el **Explorador de archivos**.
- Creamos una carpeta llamada **htop-docker** en la misma ubicación que lo estabamos haciendo con los comandos.



Abrimos

Visual Studio Code, **Bloc de notas**, o cualquier editor de texto y escribimos:

```
docker-compose.yaml ●  
C: > Windows > System32 > htop-docker > docker-compose.yaml  
1 services:  
2   htop:  
3     image: terencewestphal/htop  
4     container_name: htop_container  
5     stdin_open: true  
6     tty: true  
7     command: htop
```

Iniciar el contenedor con Docker Compose

Ejecutamos el siguiente comando para iniciar

htop en un contenedor y verificamos que el contenedor está corriendo:

```
$docker-compose up -d  
$docker ps
```

```
C:\Windows\System32\htop-docker>docker-compose up -d  
[+] Running 1/1  
✓ Container htop_container  Started  0.2s  
C:\Windows\System32\htop-docker>docker ps  
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES  
ecb37130dce7 terencewestphal/htop "htop htop" 14 seconds ago Up 14 seconds  htop_container
```

```
$docker exec -it htop_container htop
```

The screenshot shows the htop command running inside a Docker container. The top part displays system statistics: CPU usage (1%, 2%, 0%, 0%, 0%, 0%), memory usage (660M/15.6G, 0K/4.00G), tasks (2, 0, 1), load average (0.02, 0.03, 0.00), and uptime (00:30:01). The bottom part is a table showing two processes:

| PID | USER | PRI | NI | VIRT | RES | SHR | S | CPU% | MEM% | TIME+ | Command |
|-----|------|-----|----|------|------|-----|---|------|------|---------|-----------|
| 7 | root | 20 | 0 | 3668 | 1192 | 932 | R | 0.0 | 0.0 | 0:00.01 | htop |
| 1 | root | 20 | 0 | 3528 | 1228 | 976 | S | 0.0 | 0.0 | 0:00.06 | htop htop |

At the bottom, there is a blue status bar with keyboard shortcuts: F1 Help, F2 Setup, F3 Search, F4 Filter, F5 Tree, F6 Sort By, F7 Nice -, F8 Nice +, F9 Kill, F10 Quit.

Detener y eliminar el contenedor

Primero, paramos la imagen y la borramos:

The screenshot shows the Docker Hub 'Images' page for a local repository. It lists three images: mariadb (its-ubi9), hello-world (latest), and terencewestphal/htop (latest). The 'terencewestphal/htop' image has its delete icon highlighted with a red box.

| Name | Tag | Image ID | Created | Size | Actions |
|----------------------|----------|--------------|-------------|-----------|---------|
| mariadb | its-ubi9 | 90e4bcac427b | 8 days ago | 465.31 MB | |
| hello-world | latest | 74cc54e27dc4 | 22 days ago | 10.07 kB | |
| terencewestphal/htop | latest | d317eb56ee39 | 7 years ago | 6.87 MB | |

Y luego una vez paradas/borradas podemos borrar el conte

Containers [Give feedback](#)

View all your running containers and applications. [Learn more](#)

Container CPU usage [CPU usage](#)
0.05% / 1600% (16 CPUs available)

Container memory usage [Memory usage](#)
816KB / 15.19GB

Show charts

| <input type="checkbox"/> | Name | Container ID | Image | Port(s) | CPU (%) | Last started | Actions |
|--|----------------|--------------|----------------------|---------|---------|----------------|---|
| <input type="checkbox"/> | sad_galileo | 1db91595bbe9 | hello-world | | 0% | 7 days ago | ▶ ⋮ ✖ |
| <input checked="checked" type="checkbox"/> | htop-docker | | | | 0.05% | 14 minutes ago | ▶ ⋮ ✖ |
| <input type="checkbox"/> | htop_container | ecb37130dce7 | terencewestphal/htop | | 0.05% | 14 minutes ago | ▶ ⋮ ✖ |

1 2

Ejercicio 5: Imagen con Dockerfile - Aplicación Web

Creación inicial del contenedor

Descargamos la imagen base y creamos una carpeta para los archivos del sitio web con comandos:

```
$docker pull php:7.4-apache
$mkdir -p web_desplieguePelayolker
$cd web_desplieguePelayolker
```

```
C:\Windows\System32>docker pull php:7.4-apache
7.4-apache: Pulling from library/php
05e465aaa99a: Download complete
9b233e420ac7: Download complete
66d98f73acb6: Download complete
156740b07ef8: Download complete
a603fa5e3b41: Download complete
d14eb2ed1e17: Download complete
ab590b48ea47: Download complete
fb5a4c8af82f: Download complete
80692ae2d067: Download complete
fe42347c4ecf: Download complete
c428f1a49423: Download complete
25f85b498fd5: Download complete
d2c43c5efbc8: Download complete
Digest: sha256:c9d7e608f73832673479770d66aacc8100011ec751d1905ff63fae3fe2e0ca6d
Status: Downloaded newer image for php:7.4-apache
docker.io/library/php:7.4-apache

C:\Windows\System32>mkdir -p web_desplieguePelayoIker

C:\Windows\System32>cd web_desplieguePelayoIker
```

Descargamos la imagen base y creamos una carpeta para los archivos del sitio web sin comandos:

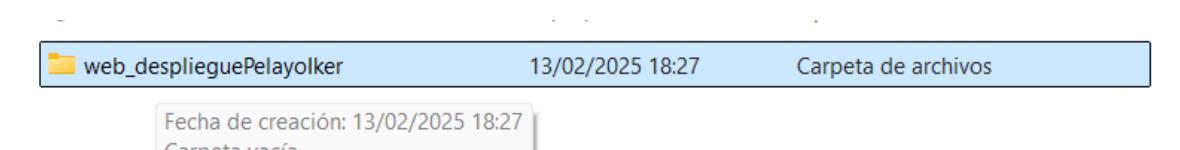
- En Docker Hub buscamos `php:7.4-apache` y descargamos la primera.
- Una vez descargada, en la pestaña "Images" (Imágenes), buscamos `php:7.4-apache`.
- Por último, creamos la carpeta manualmente y la llamamos `web_desplieguePelayolker`.



The screenshot shows the Docker Hub search results for 'php'. The first result is 'php' under the 'Docker Official Image' category. It has a star icon indicating 7.7K stars and a download icon indicating 1B+. Below the image, there's a link to 'repo-info repo's repos/php/_directory (history)' which includes '(image metadata, transfer size, etc)'. There are two bullet points: 'Image updates:' with links to 'official-images repo's library/php label' and 'official-images repo's library/php file (history)'; and 'Source of this description:' with a link to 'docs repo's php/_directory (history)'.



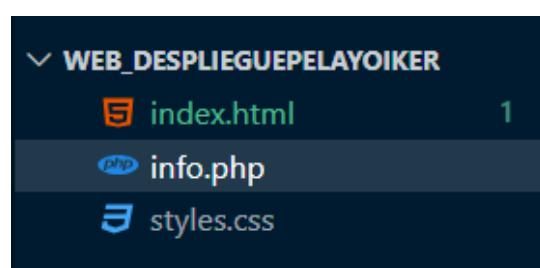
The screenshot shows the Docker desktop interface with a search bar containing 'php:7.4-apache'. The results table has columns for Name, Tag, Image ID, Created, Size, and Actions. One result is listed: 'php' with tag '7.4-apache', Image ID 'c9d7e608f738', created '2 years ago', size '647.14 MB', and actions icons.



The screenshot shows a file explorer window with a new folder named 'web_desplieguePelayolker' highlighted in blue. The folder was created on '13/02/2025 18:27' and is described as a 'Carpeta de archivos'. A tooltip 'Carpeta vacía' is visible below the folder name.

Una vez creada la carpeta, creamos los archivos del sitio.

Abrimos VS Code y creamos los archivos `index.html`, `styles.css` y `info.php`.



The screenshot shows the VS Code file explorer with a folder named 'WEB_DESPLIEGUEPELAYOIKER'. Inside the folder are three files: 'index.html', 'info.php', and 'styles.css'. The 'index.html' file has a size of 1. The 'info.php' file has a size of 1 and a PHP icon. The 'styles.css' file has a size of 1 and a CSS icon.



Archivo index.html

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Sitio Web Pelayo e Iker</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <h1>Bienvenidos al Sitio Web de Pelayo e Iker</h1>
    <p>Esta es nuestra página principal.</p>
    <p>Consulta la información del servidor <a href="info.php">aquí</a></p>
</body>
</html>
```

```
index.html X styles.css info.php
index.html > html
1  <!DOCTYPE html>
2  <html lang="es">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Sitio Web Pelayo e Iker</title>
7      <link rel="stylesheet" href="styles.css">
8  </head>
9  <body>
10     <h1>Bienvenidos al Sitio Web de Pelayo e Iker</h1>
11     <p>Esta es nuestra página principal.</p>
12     <p>Consulta la información del servidor <a href="info.php">aquí</a></p>
13  </body>
14  </html>
```



Archivo styles.css

```
body {
    font-family: 'Arial', sans-serif;
    margin: 0;
```

```
padding: 0;
background: linear-gradient(to right, #2c3e50, #4ca1af);
color: #fff;
text-align: center;
}

.container {
width: 80%;
margin: 50px auto;
background: rgba(255, 255, 255, 0.1);
padding: 20px;
border-radius: 10px;
box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.2);
}

h1 {
font-size: 2.5em;
margin-bottom: 10px;
color: #f1c40f;
}

p {
font-size: 1.2em;
line-height: 1.6;
}

@media (max-width: 768px) {
.container {
width: 90%;
}

h1 {
font-size: 2em;
}

p {
font-size: 1em;
}
}
```

The screenshot shows a code editor window with three tabs at the top: "index.html", "styles.css", and "info.php". The "styles.css" tab is active, displaying the following CSS code:

```
1 body {
2     font-family: 'Arial', sans-serif;
3     margin: 0;
4     padding: 0;
5     background: linear-gradient(to right, #2c3e50, #4ca1af);
6     color: #fff;
7     text-align: center;
8 }
9
10 .container {
11     width: 80%;
12     margin: 50px auto;
13     background: rgba(255, 255, 255, 0.1);
14     padding: 20px;
15     border-radius: 10px;
16     box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.2);
17 }
18
19 h1 {
20     font-size: 2.5em;
21     margin-bottom: 10px;
22     color: #f1c40f;
23 }
24
25 p {
26     font-size: 1.2em;
27     line-height: 1.6;
28 }
29
30 @media (max-width: 768px) {
31     .container {
32         width: 90%;
33     }
34
35     h1 {
36         font-size: 2em;
37     }
38
39     p {
40         font-size: 1em;
41     }
42 }
```

Archivo info.php

```
<?php
setlocale(LC_TIME, "es_ES.UTF-8");
$mes_actual = strftime("%B");
$fecha_actual = date("d/m/Y");
$hora_actual = date("H:i:s");

echo "<h1>Información</h1>";
echo "<p>Hoy es $fecha_actual</p>";
echo "<p>El mes es: <strong>$mes_actual</strong></p>";
echo "<p>Hora: $hora_actual</p>";
?>
```

The screenshot shows a terminal window with a dark background. At the top, there are three tabs: 'index.html' (with a file icon), 'styles.css' (with a CSS icon), and 'info.php' (with a PHP icon). The 'info.php' tab is currently active. Below the tabs, the file content is displayed with line numbers on the left:

```
1  <?php
2  setlocale(LC_TIME, "es_ES.UTF-8");
3  $mes_actual = strftime("%B");
4  $fecha_actual = date("d/m/Y");
5  $hora_actual = date("H:i:s");
6
7  echo "<h1>Información</h1>";
8  echo "<p>Hoy es $fecha_actual</p>";
9  echo "<p>El mes es: <strong>$mes_actual</strong></p>";
10 echo "<p>Hora: $hora_actual</p>";
11 ?>
```

Ahora arrancamos el contenedor al que llamamos web y que sea accesible desde un navegador en el puerto 8000.

Con comandos :

- `d` : Corre el contenedor en modo "detached" (en segundo plano).
- `-name web` : Nombre del contenedor.
- `p 8000:80` : Mapea el puerto 8000 del host al puerto 80 del contenedor.

- `v "$(pwd)":/var/www/html"`: Monta la carpeta actual en `/var/www/html` dentro del contenedor.

```
$docker run -d --name web -p 8000:80 -v C:\Users\usuario\Documents\web_des
```

```
C:\Windows\System32\web_desplieguePelayoIker>docker run -d --name web -p 8000:80 -v C:\Users\usuario\Documents\web_desplieguePelayoIker:/var/www/html php:7.4-apache
770eafe8d32cc9b5e241507c7ee7798e24d9ec4593eecda6ee84379e077dca60
```

Desde docker desktop:

- Vamos a “**Images**” y clicamos Run en la imagen de `php:7.4-apache`.
- En optional settings, lo rellenamos con los siguientes datos:
 - Container Name: web.
 - Ports: Host Port: `8000`.
 - En Volumes: **Host Path**, seleccionamos la carpeta donde guardamos nuestro sitio web(web_desplieguePelayoIker) y en Container Path(/var/www/html).
- Por último pinchamos en Run, para arrancar el contenedor.

Run a new container
php:7.4-apache

Optional settings

Container name A random name is generated if you do not provide one.

Ports

Enter "0" to assign randomly generated host ports.

Host port :80/tcp

Volumes

| | |
|--------------------------|----------------|
| Host path | Container path |
| web_desplieguePelayolker | /var/www/html |

+ Add volume

Environment variables

| | |
|----------|-------|
| Variable | Value |
|----------|-------|

+ Add environment variable

Cancel **Run**

Containers

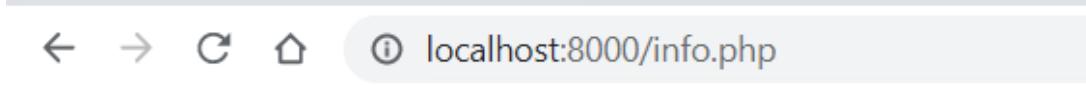
| <input type="checkbox"/> | Name | Created | Size | Actions |
|--------------------------|--------------------------|---------------|---------|---------|
| <input type="checkbox"/> | web_desplieguePelayolker | 2 minutes ago | 0 Bytes | |

Containers

| <input type="checkbox"/> | Name | Container ID | Image | Port(s) | CPU (%) | Last started | Actions |
|--------------------------|------|--------------|----------------|---------|---------|---------------|---------|
| <input type="checkbox"/> | web | 52b44c84774e | php:7.4-apache | 8080:80 | N/A | 2 minutes ago | |

Vemos la salida del script y de la página index en el navegador:

- Abrimos el navegador y entramos a <http://localhost:8000>.
- Para ver el info PHP, pinchamos en [aquí](#) o metemos este enlace <http://localhost:8000/info.php>.



Información

Hoy es 13/02/2025

El mes es: **February**

Hora: 18:38:17

Detenemos y eliminamos el contenedor con comandos

```
$docker stop web  
$docker rm web
```

```
C:\Windows\System32\web_desplieguePelayoIker>docker stop web  
web  
  
C:\Windows\System32\web_desplieguePelayoIker>docker rm web  
web
```

Detenemos y eliminamos el contenedor desde Docker Desktop

| | Name | Container ID | Image | Port(s) | CPU (%) | Last started | Actions |
|--------------------------|-------|--------------|----------------|---------|---------|---------------|--|
| <input type="checkbox"/> | ● web | 770eafe8d32c | php:7.4-apache | 8000:80 | 0% | 8 minutes ago | Stop ⋮ Delete |

Automatización de estas operaciones creando un fichero Dockerfile

Creamos un

Dockerfile en desde VS Code para construir la imagen automáticamente.

```

    ▾ WEB_DESPLIEGUEPELAYOIKER
      DockerFile
      index.html
      info.php
      styles.css

  DockerFile
  1 FROM php:7.4-apache
  2 COPY ./ /var/www/html/
  3 EXPOSE 80
  4

```

Construcción de la imagen

- Ejecutamos el siguiente comando en la carpeta donde está el `Dockerfile`.
- Verificamos la nueva imagen.

```

PS E:\CLASE\DAW\Despliegue\Docker\TareaEvaluable\web_desplieguePelayoIker> docker build -t ikerpc123/myapache2:v1 -> "E:\CLASE\DAW\DesplieguePelayoIker\dockerfile" "E:\CLASE\DAW\DesplieguePelayoIker\TareaEvaluable\web_desplieguePelayoIker"
>[1] Building 0.3s (7/7) FINISHED
-> [internal] load build definition from Dockerfile
-> [internal] transfering dockerfile: 93B
-> [internal] load history for image docker.io/library/php:7.4-apache
-> [internal] load .dockerrcignore
-> [internal] load context: 2B
-> [1/2] FROM docker.io/library/php:7.4-apache
-> [2/2] COPY ./ /var/www/html/
-> executing post-cache hooks
-> exporting to image
-> creating manifest
-> writing Image sha256:a5873ed6d396f666d573fe0216efc42a790ba4fb2ed6459a7b9b8e00eb14461
-> naming to docker.io/ikerpc123/myapache2:v1

View build details: docker-desktop/dashboard/build/desktop-linux/docker-desktop-linux/mzzn6vvsc7my0ra6ddxvyu4
PS E:\CLASE\DAW\Despliegue\Docker\TareaEvaluable\web_desplieguePelayoIker> docker images
REPOSITORY          TAG      IMAGE ID            CREATED             SIZE
mariadb             latest   a5873ed6d396        8 days ago         465MB
mariadb             lts-ubi9  90e4bccac427b   8 days ago         465MB
hello-world         latest   74cc54e27dc4   3 weeks ago        10.1kB
php                7.4-apache 20a3732f422b   2 years ago        452.5MB
terencewestphal/htop latest   d317eb56ee39   7 years ago        6.87 MB
terencewestphal/htop latest   a5873e6d5396   15 minutes ago    452.59 MB

```

```

PS E:\CLASE\DAW\Despliegue\Docker\TareaEvaluable\web_desplieguePelayoIker> docker run -d -p 8000:80 --name web_desplieguePelayoIker ikerpc123/myapache2:v1
0761297319bdcade68e63a328b6312d4fd70211b0337839b326e2d6ebcb080ac

```

| <input type="checkbox"/> | Name | Container ID | Image | Port(s) | CPU (%) | Last started | Actions |
|--------------------------|--------------------------|--------------|------------------------|-----------|---------|---------------|---------|
| <input type="checkbox"/> | web_desplieguePelayoIker | 0761297319bd | ikerpc123/myapache2:v1 | 8000:80 ↗ | 0% | 2 minutes ago | |

Images [Give feedback](#) [Q](#)

View and manage your local and Docker Hub images. [Learn more](#) [C](#)

[Local](#) [Hub repositories](#)

452.59 MB / 924.79 MB in use 5 images

| <input type="checkbox"/> | Name | Tag | Image ID | Created | Size | Actions |
|-------------------------------------|----------------------|------------|---------------|----------------|-----------|---------|
| <input type="checkbox"/> | mariadb | lts-ubi9 | 90e4bccac427b | 8 days ago | 465.31 MB | |
| <input type="checkbox"/> | hello-world | latest | 74cc54e27dc4 | 23 days ago | 10.07 kB | |
| <input type="checkbox"/> | php | 7.4-apache | 20a3732f422b | 2 years ago | 452.58 MB | |
| <input type="checkbox"/> | terencewestphal/htop | latest | d317eb56ee39 | 7 years ago | 6.87 MB | |
| <input checked="" type="checkbox"/> | ikerpc123/myapache2 | v1 | a5873e6d5396 | 15 minutes ago | 452.59 MB | |

Subimos la imagen a Docker Hub

- Nos dirigimos a “**Images**” y buscamos la imagen que subimos.
- Para subirla a Docker Hub, hacemos clic en el ícono de los tres puntos () y seleccionamos Push to Docker Hub.

Images [Give feedback](#)

View and manage your local and Docker Hub images. [Learn more](#)

Local Hub repositories

452.59 MB / 924.79 MB in use 5 Images Last refresh: 31 minutes ago

| Name | Tag | Image ID | Created |
|---------------------|-----|--------------|---------------------|
| ikerpc123/myapache2 | v1 | a5873e6d5396 | 2023-09-15 14:45:20 |

Search ☰ ⋮

View container usage
View packages and CVEs
Pull
Push to Docker Hub

Notifications

[Dismiss all](#)

✓ v1: digest:
sha256:998239460e1502281473
size: 3243

Just now ×

Desplegar la imagen en otro equipo

- Buscamos la imagen llamada ikerpc123/apache2.
- Descargamos la imagen desde Docker Hub en el otro equipo, haciendo click en “**Pull**”.

Search: ikerpc Ctrl+K ? 🔔 📦 ⚙️ ⋮ P

ikerpc ✖

Images (1) Containers (0) Volumes (0) Extensions (0) Docs (0)

Hub images (1) Hub private repos (0) Local images (0)

ikerpc123/myapache2 Tag: v1 Pull Run

to open to navigate ESC to close SCROLL for more results Give feedback

- Comprobamos que esta entre nuestras imágenes.

Streamlined Plans. [Learn more](#)

Images [Give feedback](#)

View and manage your local and Docker Hub images. [Learn more](#)

[Local](#) [Hub repositories](#)

734.43 MB / 1.04 GB In use 7 images

Search

| <input type="checkbox"/> | Name | Tag | Created | Size | Actions | Image ID |
|--------------------------|--------------------------------------|--------|----------------|-----------|---|--------------|
| <input type="checkbox"/> | nginx | latest | 8 days ago | 191.99 MB | ▶ ⋮ trash | 97662d24417b |
| <input type="checkbox"/> | hello-world | latest | 23 days ago | 10.07 kB | ▶ ⋮ trash | 74cc54e27dc4 |
| <input type="checkbox"/> | portainer/portainer-docker-extension | 2.21.5 | 2 months ago | 307.87 MB | ▶ ⋮ trash | a1f140f40753 |
| <input type="checkbox"/> | nicolaka/netsniff | latest | 9 months ago | 542.43 MB | ▶ ⋮ trash | 27b858cdcd8a |
| <input type="checkbox"/> | docker/disk-usage-extension | 0.2.9 | 1 year ago | 2.81 MB | ▶ ⋮ trash | 0f1db5eed760 |
| <input type="checkbox"/> | docker/logs-explorer-extension | 0.2.6 | 1 year ago | 12.13 MB | ▶ ⋮ trash | 39b97f06b988 |
| <input type="checkbox"/> | ikerp123/myapache2 | v1 | 25 minutes ago | 452.59 MB | ▶ ⋮ trash | a5873e6d5396 |

PERSONAL
pelayus

Account Settings
Upgrade
1 hour ago [↻](#)
Sign out