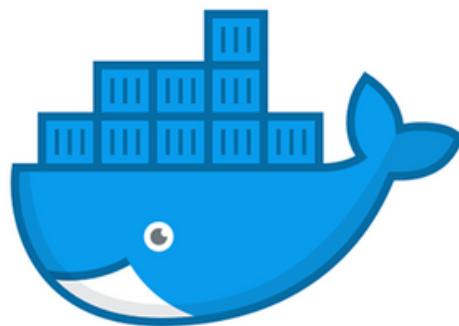




TAREA EVALUABLE

DOCKER



Andrea Gómez Fueyo
Sandra Rujas Arroyo

ACTIVIDAD EVALUABLE DE DOCKER

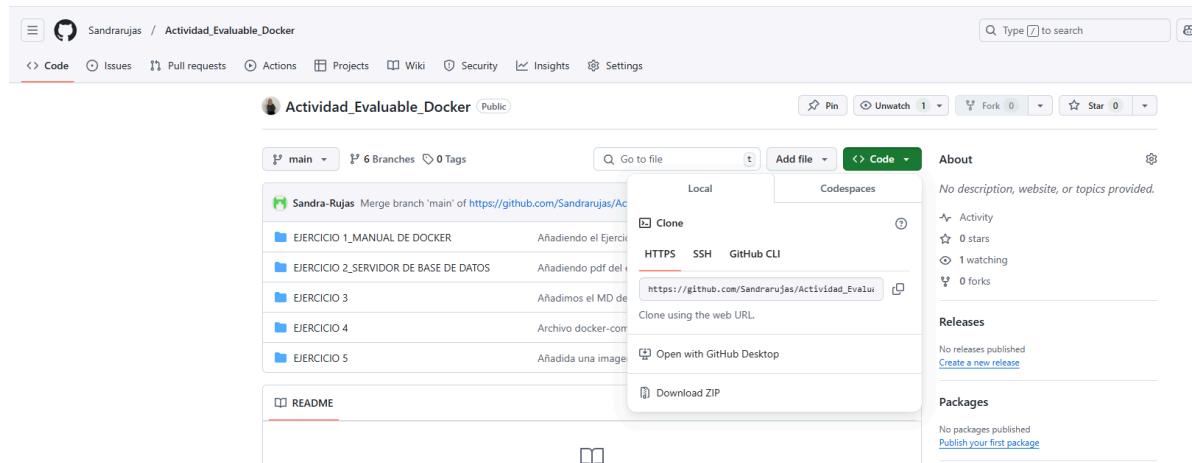
ACTIVIDAD EVALUABLE DE DOCKER

- Enlace al repositorio.
- Integrantes.
- Ramas.
- Desarrollo de las tareas y plazos.

- Enlace al repositorio.

En el siguiente enlace encontrarás el repositorio de Github donde se encuentran los diferentes ejercicios:

https://github.com/Sandrarujas/Actividad_Evaluabe_Docker.git

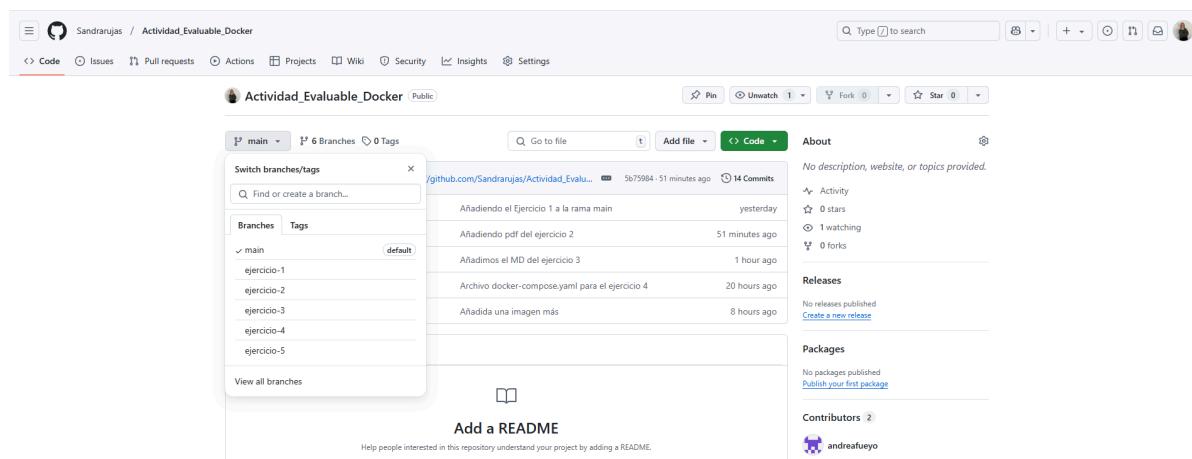


- Integrantes.

El grupo está formado por **Andrea Gómez Fueyo** y **Sandra Rujas Arroyo**. En este caso, el jefe de equipo ha sido Sandra Rujas.

- Ramas.

Para el desarrollo de este ejercicio, hemos utilizado seis ramas diferentes, una rama por ejercicio:



- **Rama Main:** nuestra rama principal. Dentro de esta rama encontrarás el conjunto de ejercicios al completo.

- **Rama "ejercicio-1":** rama que contiene el ejercicio 1 completo. Dentro de esta rama encontrarás una carpeta con las imágenes, una un documento `.md` y `.pdf` con el contenido de la actividad.
- **Rama "ejercicio-2":** rama que contiene el ejercicio 2 completo. Dentro de esta rama encontrarás una carpeta con las imágenes, una un documento `.md` y `.pdf` con el contenido de la actividad.
- **Rama "ejercicio-3":** rama que contiene el ejercicio 3 completo. Dentro de esta rama encontrarás una carpeta con las imágenes, una un documento `.md` y `.pdf` con el contenido de la actividad.
- **Rama "ejercicio-4":** rama que contiene el ejercicio 4 completo. Dentro de esta rama encontrarás una carpeta con las imágenes, una un documento `.md` y `.pdf` con el contenido de la actividad.
- **Rama "ejercicio-5":** rama que contiene el ejercicio 5 completo. Dentro de esta rama encontrarás una carpeta con las imágenes, una un documento `.md` y `.pdf` con el contenido de la actividad.

- Desarrollo de las tareas y plazos.

En cuanto al desarrollo de tareas, el reparto ha sido de la siguiente forma:

- **Ejercicio 1 - Manual de docker:** Sandra Rujas Arroyo
- **Ejercicio 2 - Servidor de base de datos:** Sandra Rujas Arroyo
- **Ejercicio 3 - Contenedores en red:** Andrea Gómez Fueyo
- **Ejercicio 4 - Docker Compose:** Andrea Gómez Fueyo
- **Ejercicio 5 - Imagen con Dockerfile - Aplicación web:** Andrea Gómez Fueyo y Sandra Rujas Arroyo

En este caso, para el desarrollo de las tareas, no hemos definido un plazo concreto. Cada integrante iba subiendo sus tareas según pudiese. Por tanto, no había tiempo estimado para el desarrollo de las mismas. Simplemente había que subirlas al repositorio antes del tiempo estipulado por la profesora en el aula virtual.

Resultado final: se puede encontrar en el repositorio añadido en el punto de "*Enlace al repositorio*". Se explicará de manera detallada en la exposición.

Mayores dificultades: leves faltas de comunicación en una de las tareas, pero por lo demás, cada una ha realizado su trabajo de manera satisfactoria.

Beneficios de trabajar en grupo: Trabajar en pareja tiene muchas ventajas en comparación con hacerlo solo. Al colaborar con otra persona, podemos combinar nuestras habilidades y conocimientos, lo que nos ayuda a ser más creativos y resolver problemas de manera más eficiente. Además, dividir las tareas permite que el trabajo sea más rápido y menos pesado para cada uno. También es útil recibir retroalimentación inmediata, ya que esto nos ayuda a corregir errores y mejorar la calidad del resultado final. Otro beneficio importante es el apoyo mutuo, que reduce el estrés y aumenta la motivación. En general, trabajar en pareja mejora la comunicación, el aprendizaje y nos permite obtener mejores resultados.

EJERCICIO 1- MANUAL DE DOCKER

Realizado por Andrea Gómez Fueyo y Sandra Rujas Arroyo

EJERCICIO 1- MANUAL DE DOCKER

1. Instalación de Docker Desktop

1.1 Requisitos del Sistema

1.2 Pasos de Instalación

2. Configuración Inicial

3. Interfaz de Docker Desktop

3.1. Dashboard (Tablero de control)

3.2. Images (Imágenes)

3.3. Containers (Contenedores)

3.4. Volumes (Volúmenes)

3.5. Builds (Construcciones)

3.6. Networks (Redes)

3.7. Extensions (Extensiones)

3.8. Settings (Configuración)

3.8.1.General

3.8.2. Resources (Recursos)

3.8.3. Docker Engine

3.8.4. Kubernetes

3.8.5. Extensions (Extensiones)

3.8.6. Updates (Actualizaciones)

3.8.7. Troubleshoot (Solucionar problemas)

3.8.8. Experimental Features (Funciones Experimentales)

4. Otras áreas y secciones:

4.1. Docker Home:

4.2. Docker Admin Console:

4.3. Docker Hub:

4.4. Docker Scout:

4.5. Docker Build Cloud:

4.6. Testcontainers Cloud:

5. Resumen final de las funcionalidades clave de Docker Desktop

5.1. Descargar e instalar Docker Desktop

5.2. Usar Docker Desktop

5.3. Configuración de recursos

5.4. Crear y ejecutar contenedores

5.5. Docker Compose

5.6. Terminal integrada

5.7. Integración con Docker Hub

5.8. Docker Desktop en WSL 2 (Windows)

Docker Desktop es una aplicación que permite a los desarrolladores construir, compartir y ejecutar aplicaciones en contenedores de manera sencilla. Este manual proporciona una guía básica para su instalación, configuración y uso.

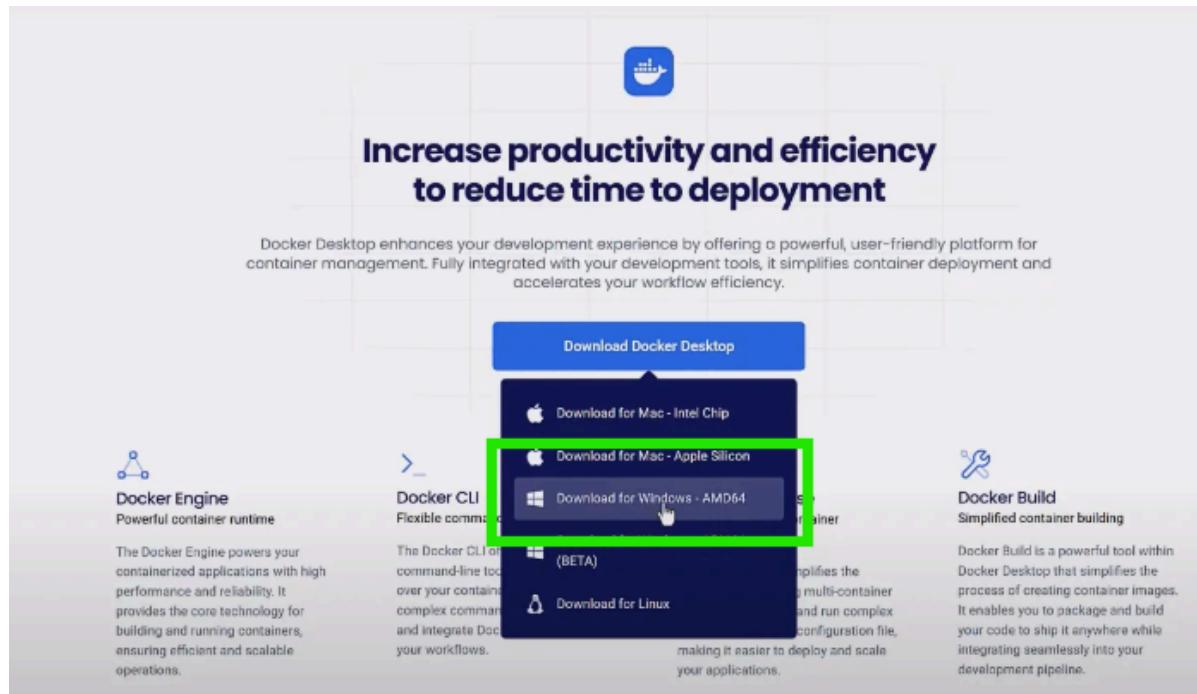
1. Instalación de Docker Desktop

1.1 Requisitos del Sistema

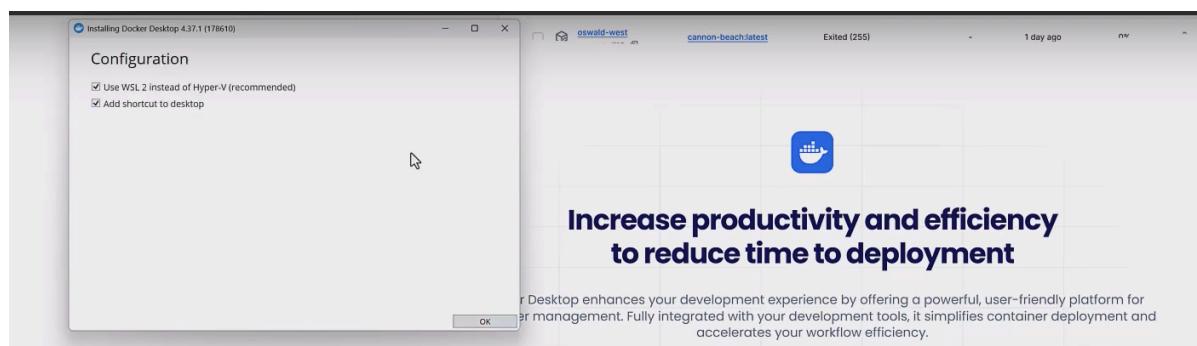
- Windows 10/11 (con WSL2 habilitado) o macOS 10.14+
- Procesador de 64 bits con soporte para virtualización
- Al menos 4 GB de RAM

1.2 Pasos de Instalación

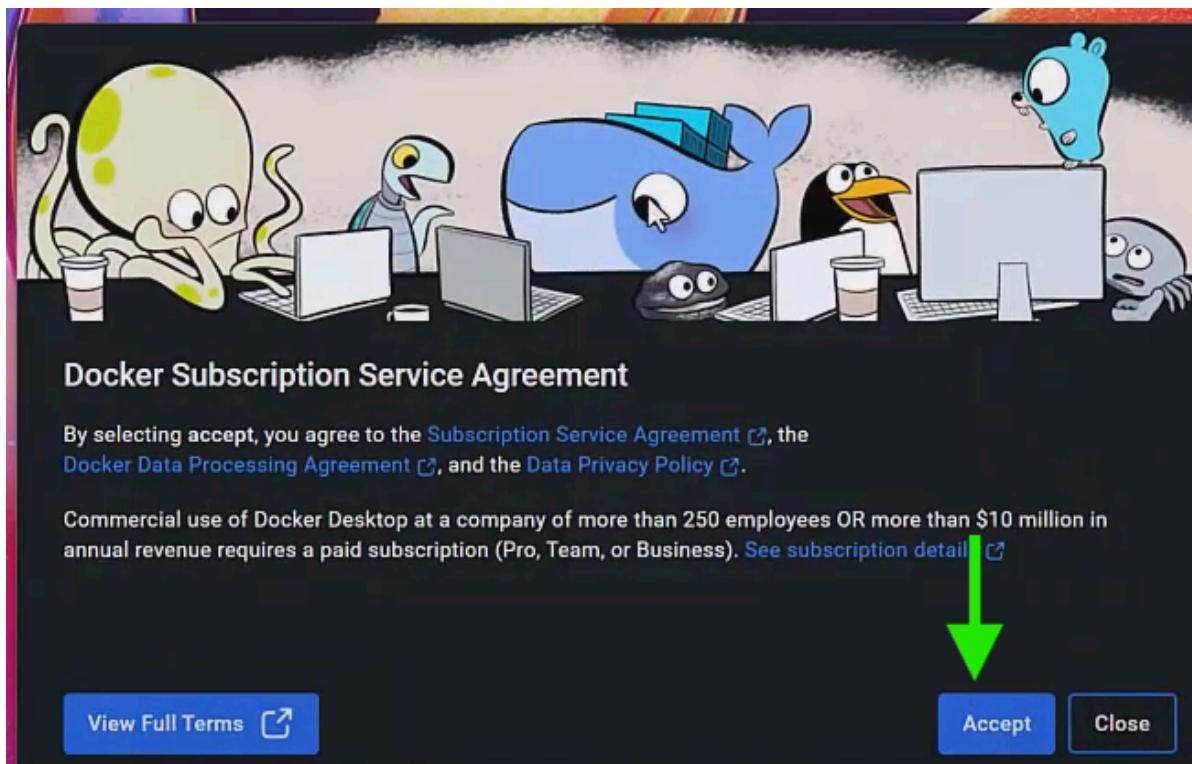
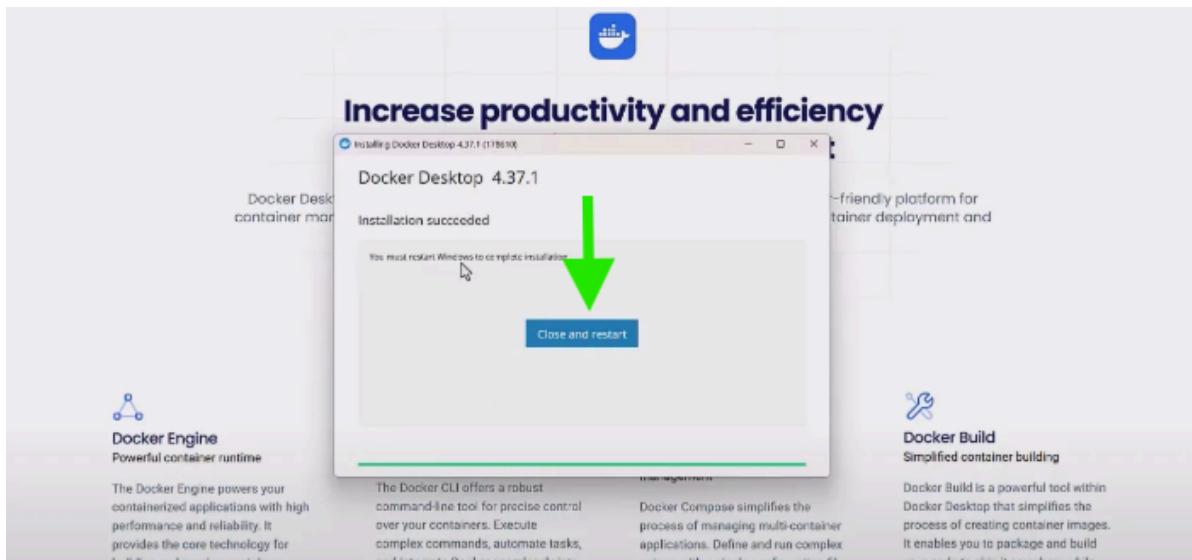
Descargamos Docker Desktop desde el sitio oficial: <https://www.docker.com/products/docker-desktop>.



Ejecutamos el instalador y seguimos las instrucciones en pantalla.



Reiniciamos el sistema si es necesario.



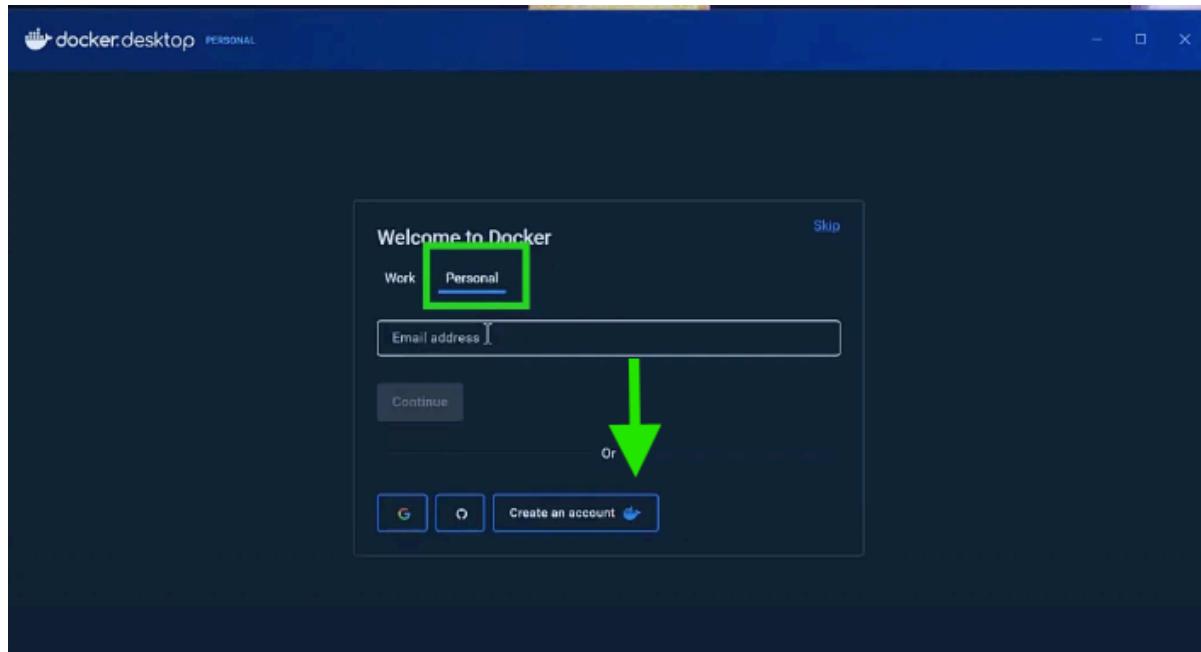
Iniciamos Docker Desktop y verificamos que esté corriendo desde la barra de tareas o con el comando:

```
docker --version
```

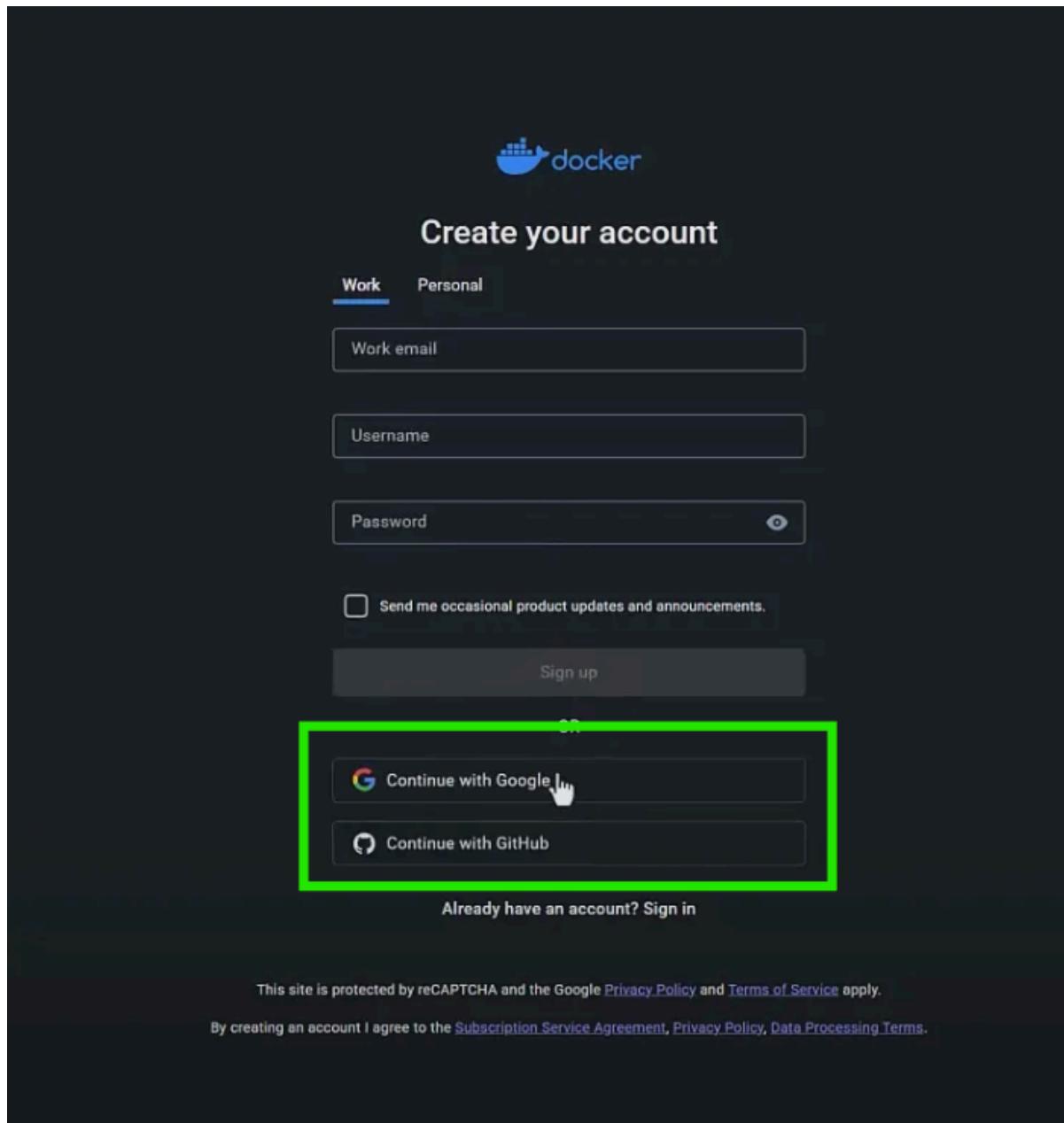
```
C:\Users\34615>docker --version
Docker version 25.0.3, build 4debf41
```

2. Configuración Inicial

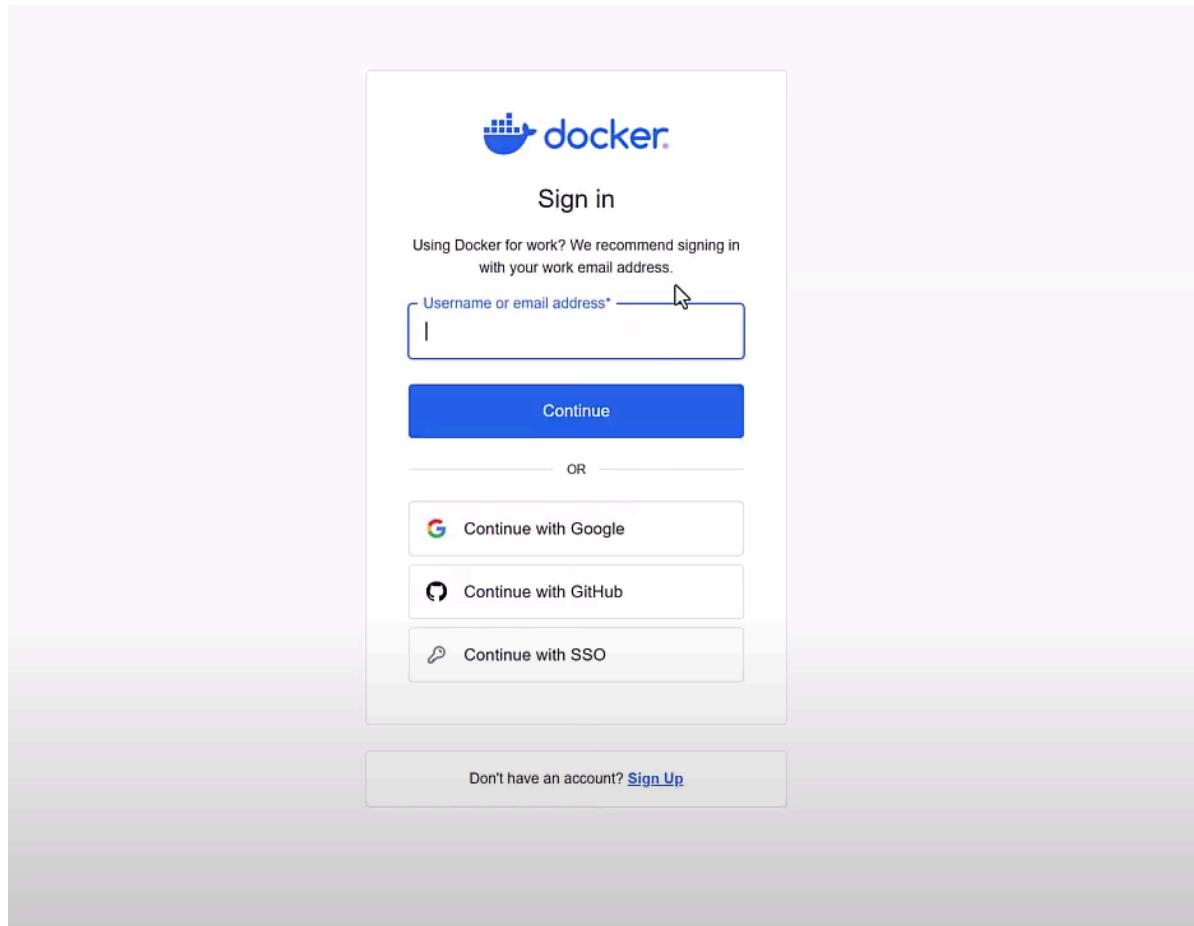
Configuramos Docker, le damos uso de manera “Personal”. Si ya tenemos una cuenta de Docker, debemos introducirla en la casilla de “Email address”.



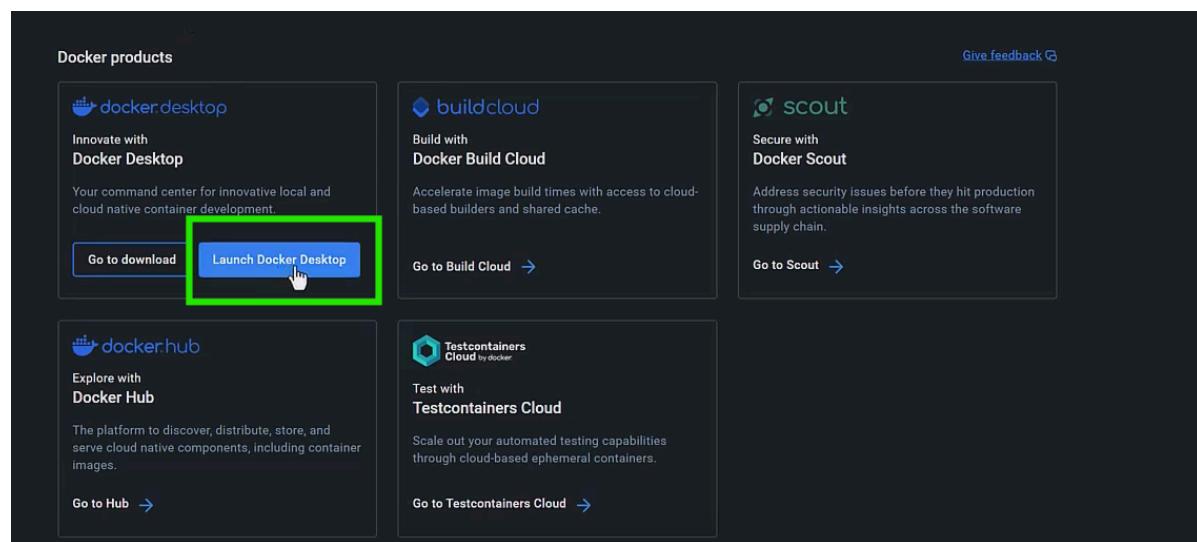
Si por el contrario no tenemos, deberemos darle a la casilla de “*Create an account*” para crearnos una cuenta nueva. Tenemos la opción de poder utilizar nuestra cuenta de *Google* o *Github*, si no queremos crear una cuenta desde cero.

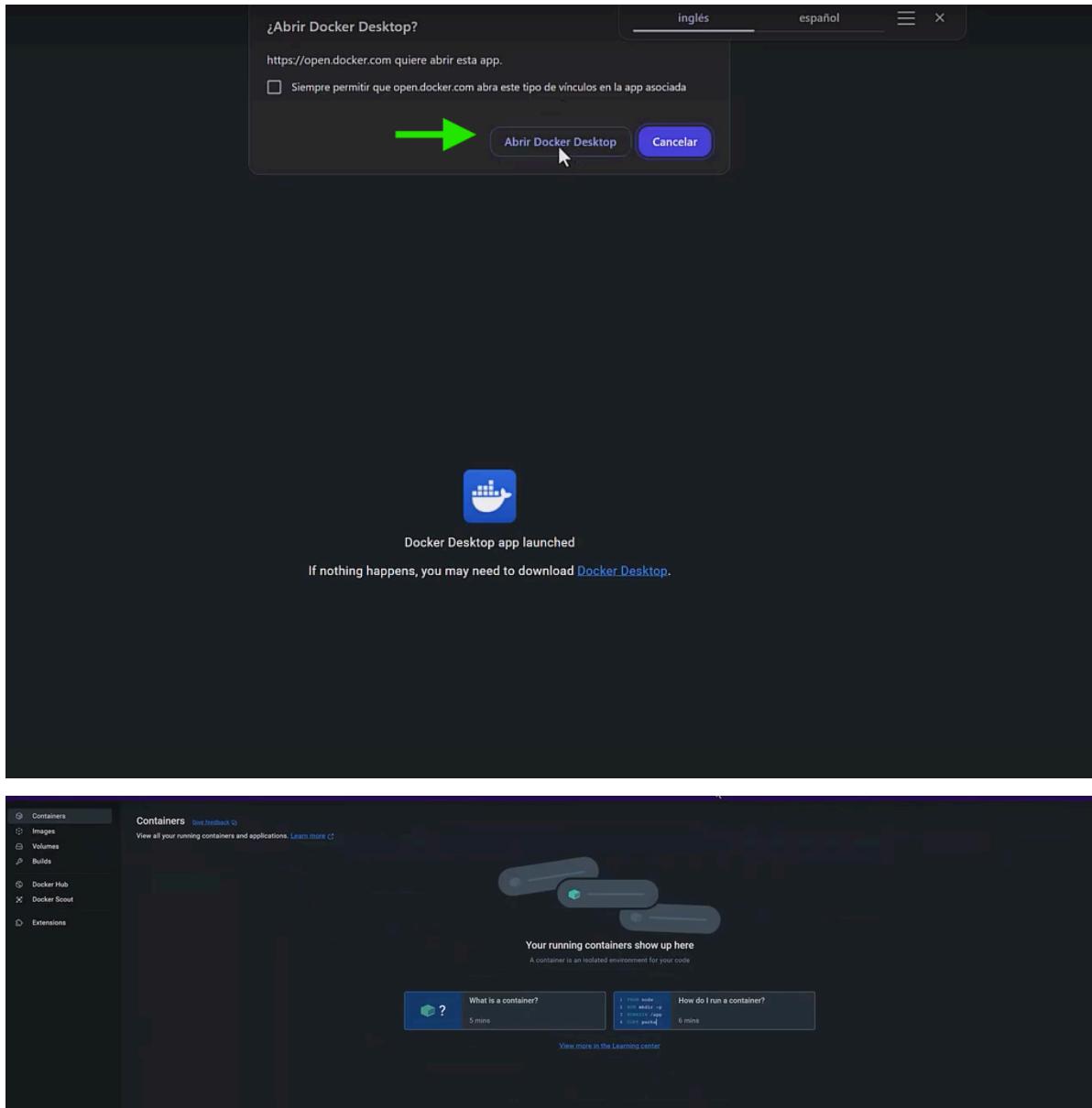


Pongamos que si tenemos una cuenta creada, pulsamos la tecla de *Sign in* que podemos observar en la imagen anterior y a continuación, nos saldrá la siguiente imagen:



Una vez entramos en nuestra cuenta, podemos lanzar el Docker Desktop:





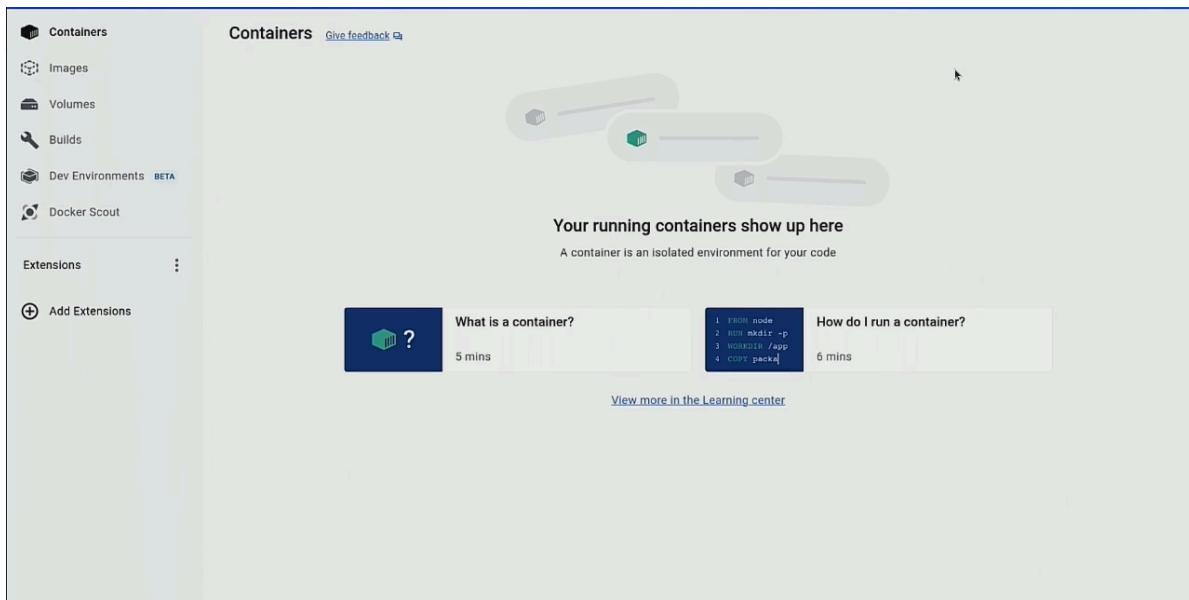
3. Interfaz de Docker Desktop

Cuando abres **Docker Desktop**, verás varias secciones clave:

3.1. Dashboard (Tablero de control)

Página principal donde puedes visualizar y gestionar tus contenedores, imágenes, volúmenes y redes. Aquí puedes:

- **Ver contenedores activos y detenidos:** Muestra una lista de contenedores en ejecución, con información como su estado, nombre, imagen y puertos expuestos.
- **Iniciar y detener contenedores:** Puedes iniciar, detener y eliminar contenedores desde esta interfaz.
- **Visualizar logs:** Para ver el registro de eventos de los contenedores y facilitar la depuración.

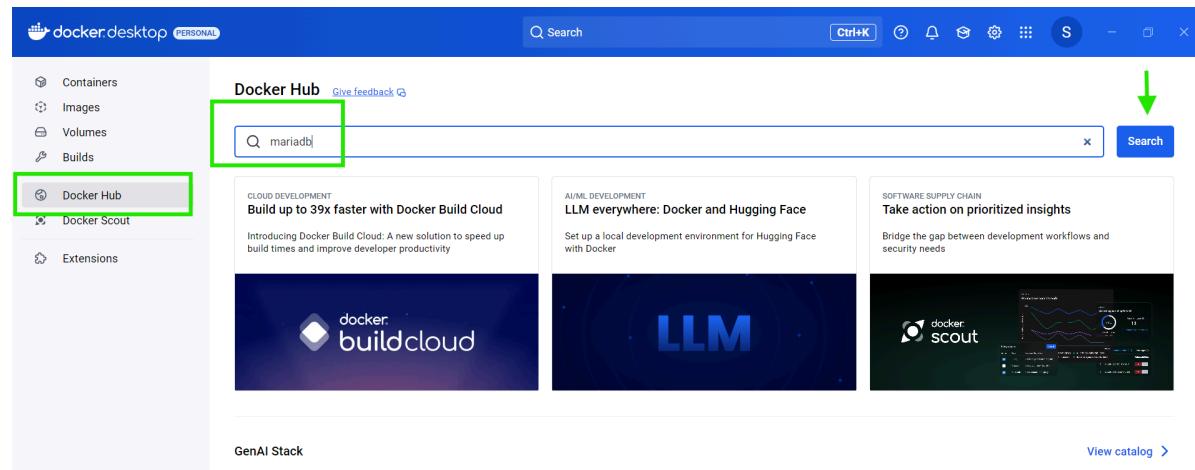


3.2. Images (Imágenes)

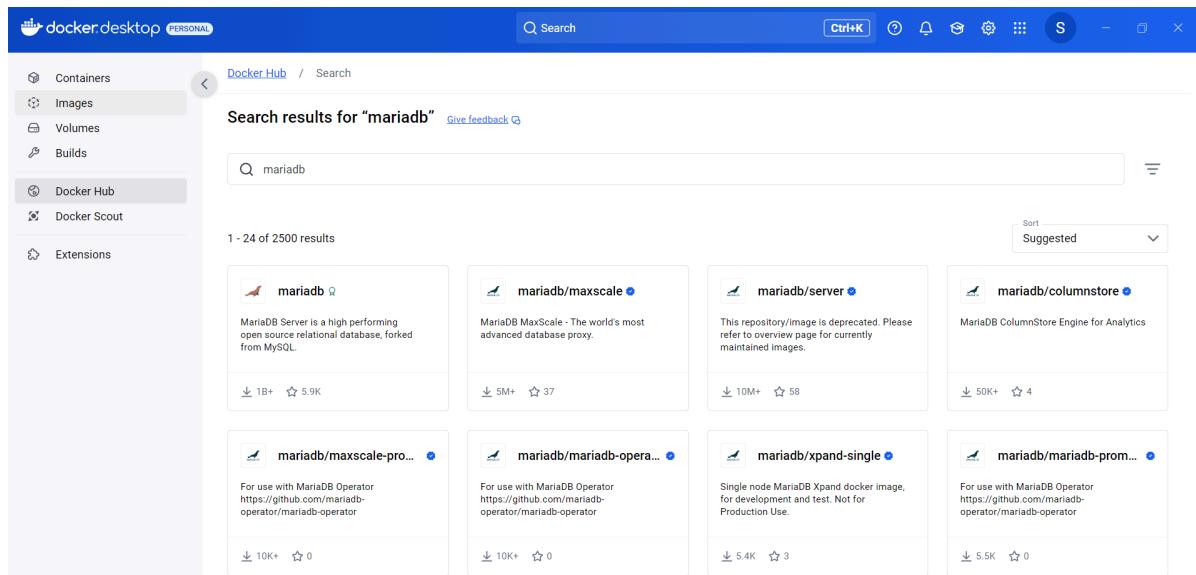
A través del área de *imágenes*, podemos gestionar todas las **imágenes Docker**. Las imágenes son plantillas para crear contenedores. Desde esta sección puedes:

- **Buscar y descargar imágenes:** Con el botón "Pull", puedes buscar imágenes oficiales o personalizadas desde Docker Hub y otras fuentes.

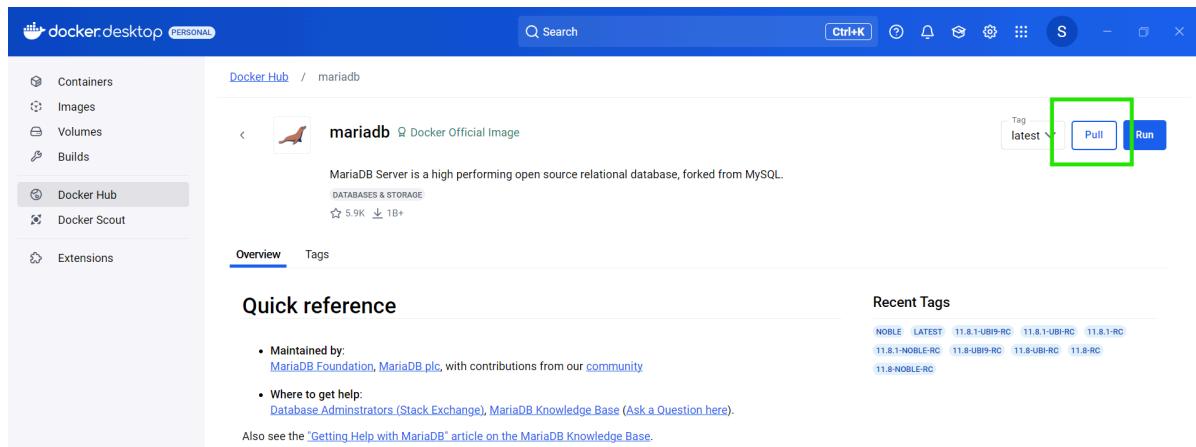
Ejemplo práctico: Vamos al área de Docker hub y buscamos la imagen que queramos descargar. En nuestro ejemplo es la última imagen de mariadb:



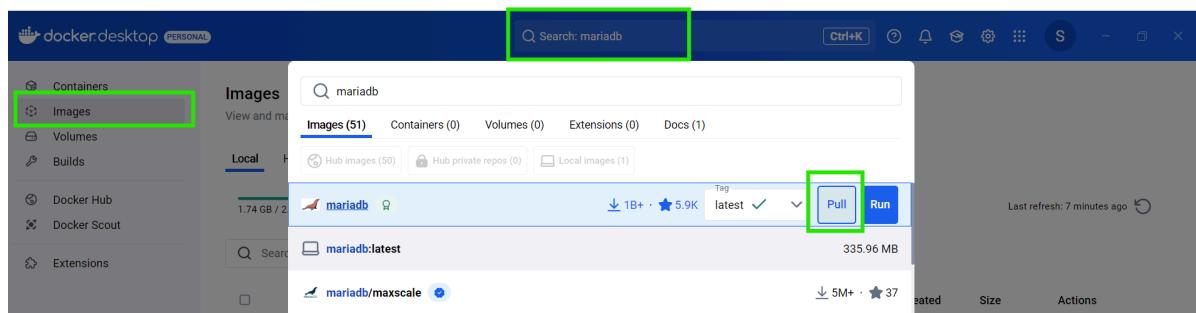
Nos aparecen las diferentes imágenes que existen con dicho nombre:



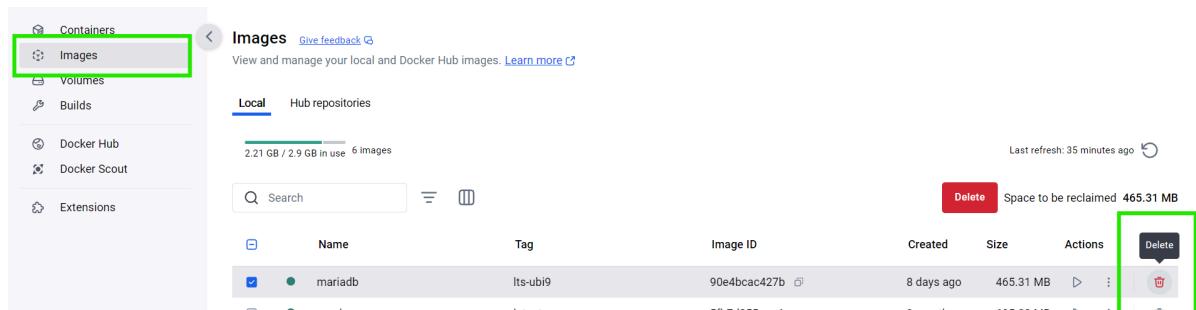
Seleccionamos la oficial y clickamos en *Pull* para descargarla:



También podemos hacerlo a través el área de imágenes. Pulsamos sobre el buscador "Search" y escribimos el nombre de la imagen que queramos descargar. En nuestro caso hemos puesto "mariadb", por lo que nos aparecerá la última imagen. Seguidamente daríamos click en *RUN* y se descargaría:



- Eliminar imágenes:** Puedes eliminar imágenes que no estés utilizando, ya que si dicha imagen está en uso, no será posible su eliminación.



- **Ver detalles:** Información detallada sobre cada imagen, como el tamaño y las etiquetas.

The screenshot shows the Docker Hub page for the **mariadb** image. At the top, there's a search bar, a sign-in button, and a "Sign up" button. Below the header, the image name **mariadb** is displayed with a seal icon, followed by "Docker Official Image", "1B+", and "5.8K". A brief description states: "MariaDB Server is a high performing open source relational database, forked from MySQL." To the right, there are buttons for "docker pull mariadb" and "Copy". Below this, there are tabs for "Overview" and "Tags", with "Tags" being the active one. Under "Tags", there are filters for "Sort by" (set to "Newest"), "Filter Tags", and a search bar. The main table lists four tags under the "latest" tag heading, each with a "Digest" (a long hash), "OS/ARCH", "Vulnerabilities" (a grid of colored boxes showing counts for different severity levels), and "Compressed Size".

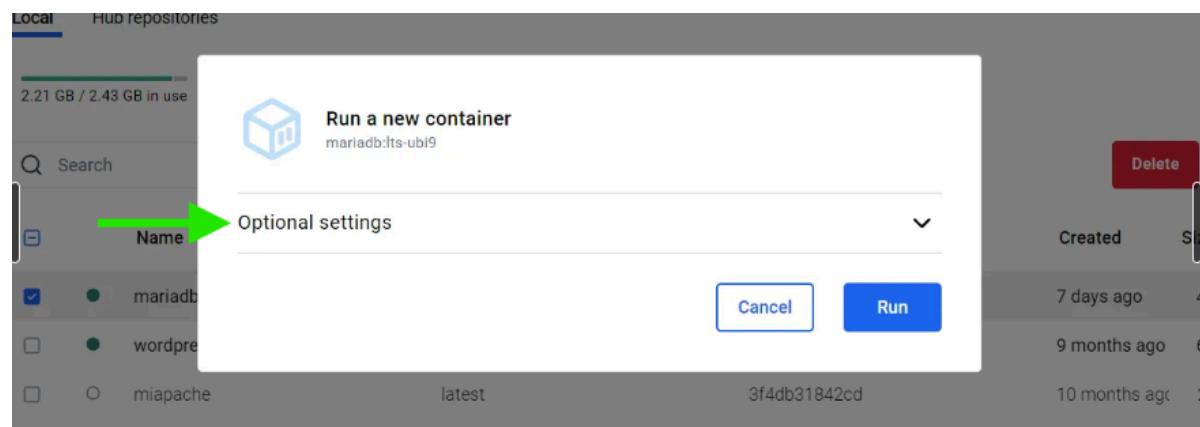
Digest	OS/ARCH	Vulnerabilities	Compressed Size
9586c56a4e03	linux/amd64	3 32 14 6 5	118.83 MB
5dde1f996e76	linux/arm64	3 32 14 6 5	116.79 MB
89e47d5278dd	linux/ppc64le	3 32 14 6 5	129.33 MB
f78583006cc9	linux/s390x	3 32 14 6 5	124.99 MB

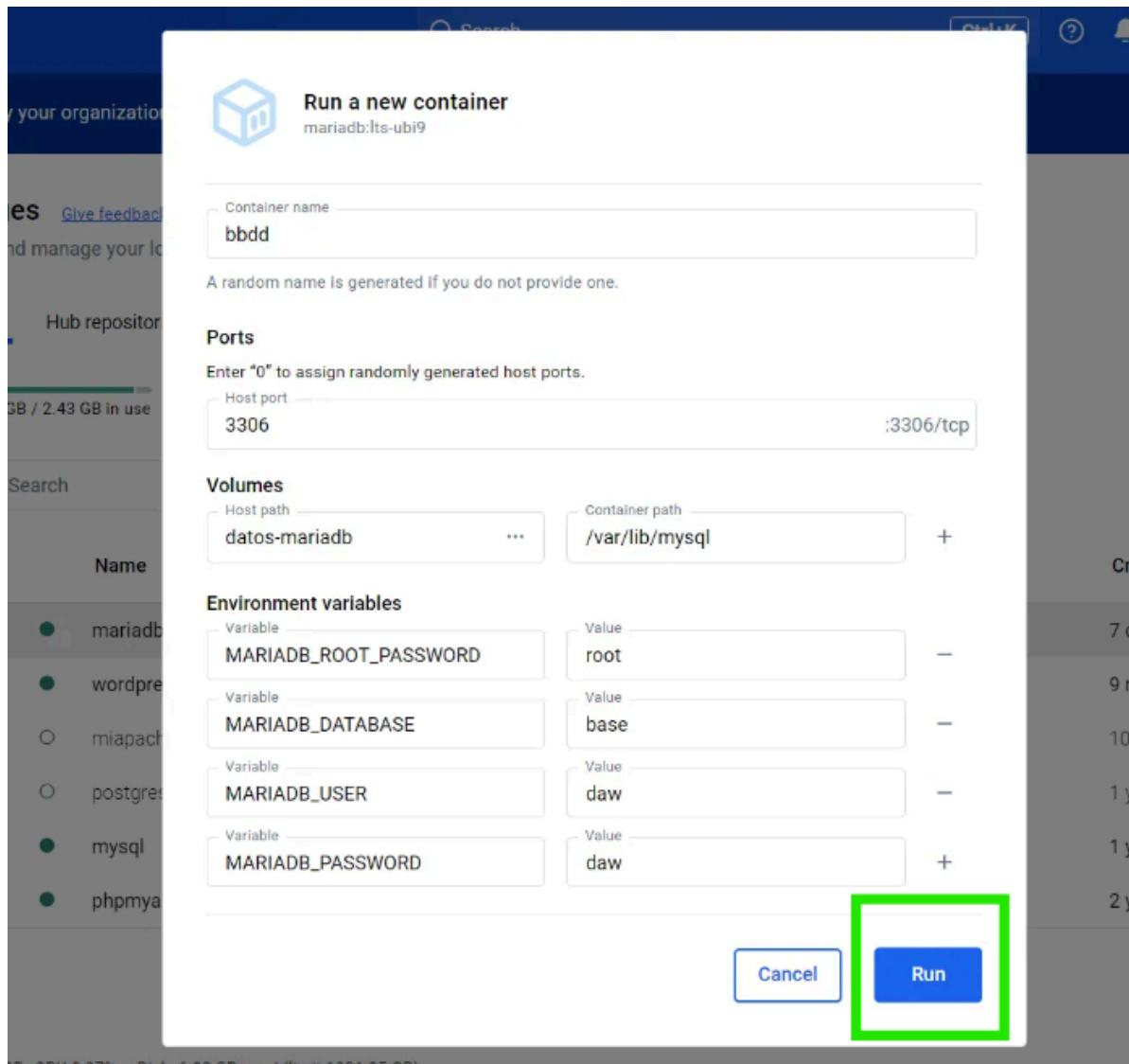
3.3. Containers (Contenedores)

Desde aquí puedes:

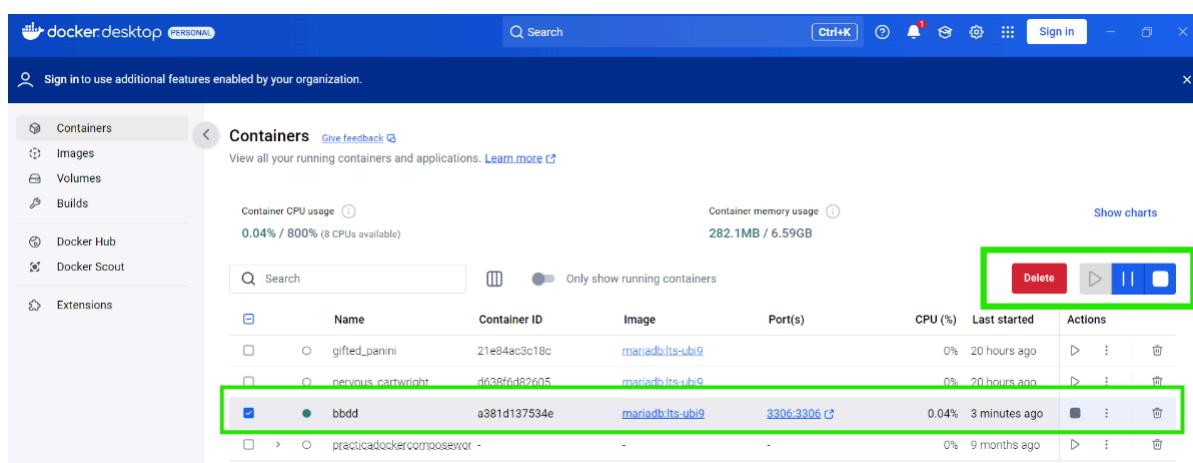
- **Gestionar contenedores:** Configurar, detener, eliminar, pausar y reiniciar contenedores. Puedes interactuar con los contenedores sin tener que usar la terminal.

Si pulsamos en *Optional settings*, podremos configurar los datos de nuestro contenedor:

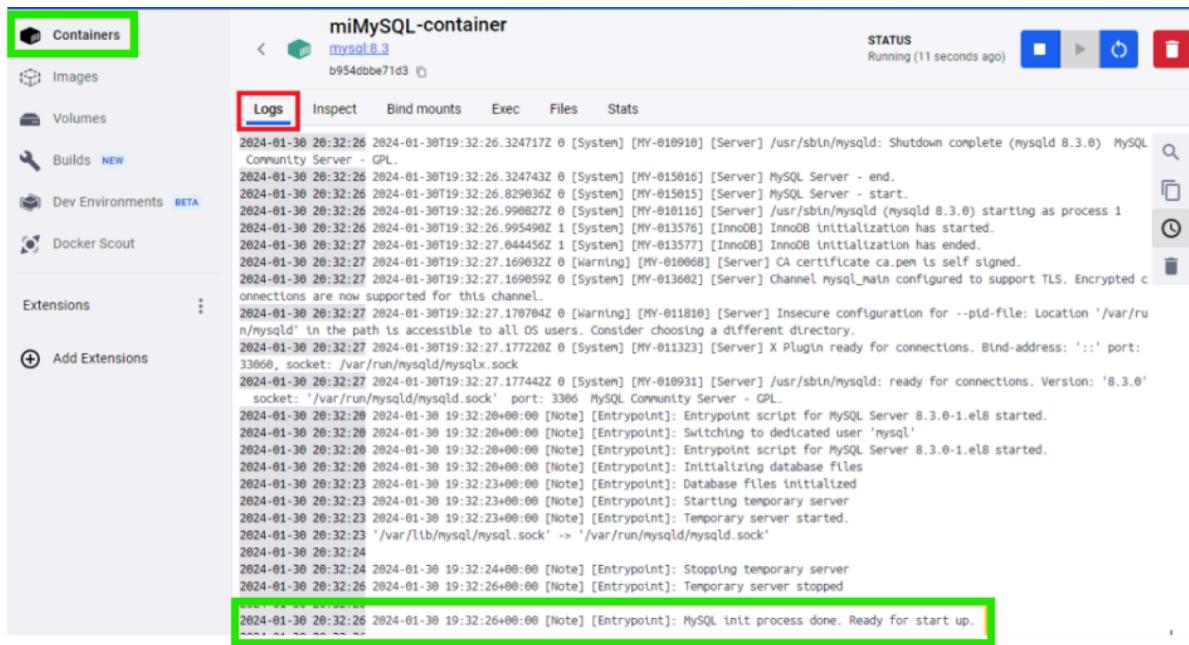




Una vez introducidos los datos, pulsamos en *RUN* para su creación. En el área de "containers" te aparecerá lo siguiente:



- **Ver contenedores en ejecución:** Además de las acciones básicas, puedes ver los logs de cada contenedor, lo cual es útil para depurar. Los logs son los registros de salida generados por la aplicación que se ejecuta dentro de él. Estos registros incluyen mensajes de error, información de depuración, eventos del sistema y cualquier otro tipo de salida que la aplicación escriba en la consola estándar o de error.



3.4. Volumes (Volúmenes)

Los volúmenes son espacios de almacenamiento persistentes fuera de los contenedores. Desde esta sección puedes:

- Gestionar volúmenes:** Ver los volúmenes existentes, eliminarlos o crear nuevos.

Volumes [Give feedback](#)

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

New Volume

Name your volume

MiVolumenDocker

Cancel Create

- Usar volúmenes para persistencia de datos:** Puedes adjuntar volúmenes a contenedores para mantener los datos entre reinicios de contenedores.



Run a new container

mariadb:lts-ubi9

Container name

bbdd

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

MARIADB_ROOT_PASSWORD

Value

root

-

Variable

MARIADB_DATABASE

Value

base

-

Variable

MARIADB_USER

Value

daw

-

Variable

MARIADB_PASSWORD

Value

daw

+

Cancel

Run

Este apartado de volúmenes que he seleccionado en la imagen anterior, se encuentra en la configuración de los contenedores, (*apartado "3.3 Containers"*). A continuación, muestro como, después de haber introducido los datos de configuración del contenedor junto con su volumen (con nombre: *datos-mariadb*), aparece en nuestra sección o área de volúmenes:

The screenshot shows the Docker interface with the 'Volumes' section highlighted. The 'Volumes' section lists several volumes, including 'datos-mariadb', which is highlighted with a green box. The table columns are 'Name', 'Created', 'Size', and 'Actions'. The 'datos-mariadb' volume was created 50 minutes ago, has a size of 166.1 MB, and has edit and delete icons.

Name	Created	Size	Actions
0503e8fad30203f067e35be61d50f957c3faed80ca5b5a751dc48cc4976ada6b	10 months ago	45.9 MB	
3dd453fb4a93a1d22b8c15dabfc06714bed4207c6a605df63f6f2c8e11bdc6c1	10 months ago	45.5 MB	
4ee9cf94691bbd55bdb02ba700f3e6523592b995c5a589db04ff6a71c81b1d5	10 months ago	45.4 MB	
a25398a89e58896dcc6d21fa64af3eb3f725a7f3de4ec85ea5584d5888b65d04	21 hours ago	0 Bytes	
a70d467000bcb70ffbd9d1e001a9e55d0b411a64f727-20bbabb423c76d0d0	21 hours ago	0 Bytes	
datos-mariadb	50 minutes ago	166.1 MB	
1ac03c70e649797c10a81487c13034520607703189728cc3a2a539c3	10 months ago	45.7 MB	

3.5. Builds (Construcciones)

El apartado **Builds** en Docker Desktop te permite construir imágenes **directamente desde la interfaz gráfica** sin tener que recurrir a la línea de comandos, aunque también puedes hacerlo desde la terminal usando el comando `docker build`.

ID	Name	Builder	Duration	Created	Author
02jtpq	docker2	default	1m 30s	10 months ago	N/A
f3gdnx	docker2	default	0.5s	10 months ago	N/A
8cluel	docker2	default	1.7s	10 months ago	N/A
4smucb	docker2	default	1m 22s	10 months ago	N/A
j7fc6	docker2	default	0.1s	10 months ago	N/A
r2l85t	docker2	default	0.1s	10 months ago	N/A
q8w01g	docker2	default	43.1s	10 months ago	N/A
uo2d1u	docker2	default	0.0s	10 months ago	N/A
p5hs8l	docker2	default	0.1s	10 months ago	N/A

Con **Builds** en Docker Desktop, puedes:

- 1. Crear nuevas imágenes:** Desde la interfaz, puedes crear una nueva imagen a partir de un **Dockerfile** que hayas preparado en tu proyecto. Esto es útil si estás trabajando en una aplicación y necesitas crear una imagen personalizada.
- 2. Verificar el proceso de construcción:** Docker Desktop te muestra el progreso de la construcción de la imagen. Puedes ver qué pasos se están ejecutando (como la instalación de paquetes, la copia de archivos, etc.) y si ocurre algún error durante el proceso.
- 3. Usar archivos Dockerfile existentes:** Si ya tienes un **Dockerfile** en tu proyecto, puedes seleccionarlo directamente en Docker Desktop y usarlo para crear la imagen. No es necesario escribir comandos complejos en la terminal para ejecutar un `docker build`.
- 4. Gestionar múltiples Build Contexts:** Puedes especificar qué directorios o archivos deben ser parte del contexto de la construcción. Un *Build Context* es el directorio que Docker usa para acceder a los archivos necesarios para construir la imagen. Esto incluye el **Dockerfile** y los archivos que serán copiados a la imagen durante el proceso de construcción.

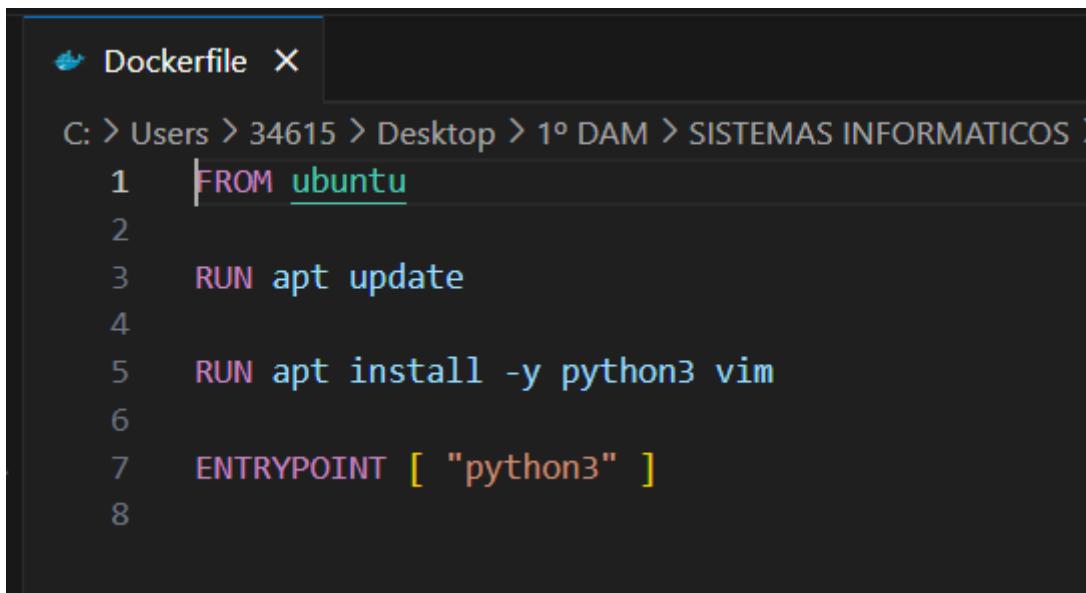
Pero, ¿qué es un **Dockerfile**?

Un **Dockerfile** es un archivo de texto que contiene un conjunto de instrucciones para construir una imagen de Docker de forma automatizada. Básicamente, define qué sistema operativo, dependencias, configuraciones y archivos debe incluir la imagen para que el contenedor funcione correctamente.

¿Para qué sirve un **Dockerfile**?

- Automatiza la creación de imágenes de Docker
- Garantiza entornos reproducibles en cualquier sistema
- Facilita la implementación y escalabilidad de aplicaciones
- Permite personalizar imágenes base agregando configuraciones específicas

Ejemplo:



```
C: > Users > 34615 > Desktop > 1º DAM > SISTEMAS INFORMATICOS >
1  FROM ubuntu
2
3  RUN apt update
4
5  RUN apt install -y python3 vim
6
7  ENTRYPOINT [ "python3" ]
8
```

¿Que significa este Dockerfile?

Este Dockerfile define una imagen basada en **Ubuntu** con **Python 3** y **Vim** instalados. Vamos a desglosarlo línea por línea:

`FROM ubuntu`

- Especifica que la imagen base es **Ubuntu** (probablemente la última versión disponible en Docker Hub).

`RUN apt update`

- Ejecuta `apt update` para actualizar la lista de paquetes disponibles en el sistema.

`RUN apt install -y python3 vim`

- Instala **Python 3** y **Vim** sin pedir confirmación (`-y`).

`ENTRYPOINT ["python3"]`

- Define que, al ejecutar un contenedor basado en esta imagen, el comando predeterminado será `python3`.
- Esto significa que, si ejecutas el contenedor sin argumentos, entrarás directamente en el intérprete de Python.

Ejemplo de uso:

Si construimos la imagen con:

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

Y luego ejecutamos un contenedor con:

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

Entraremos directamente en la consola interactiva de **Python 3**.

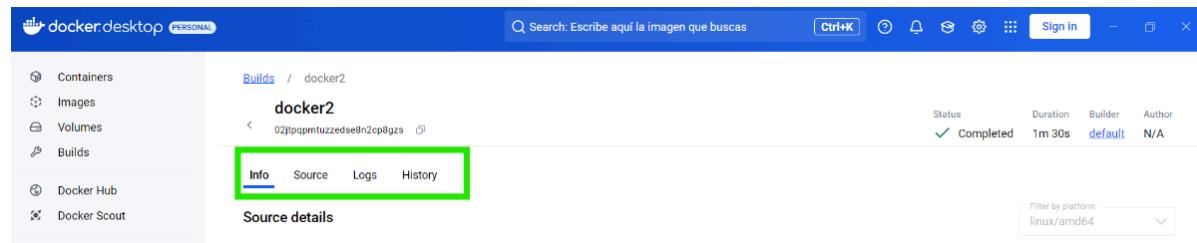
Si queremos ejecutar un script Python al iniciar el contenedor, deberemos ejecutar el siguiente comando:

```
$docker run mi-imagen script.py
```

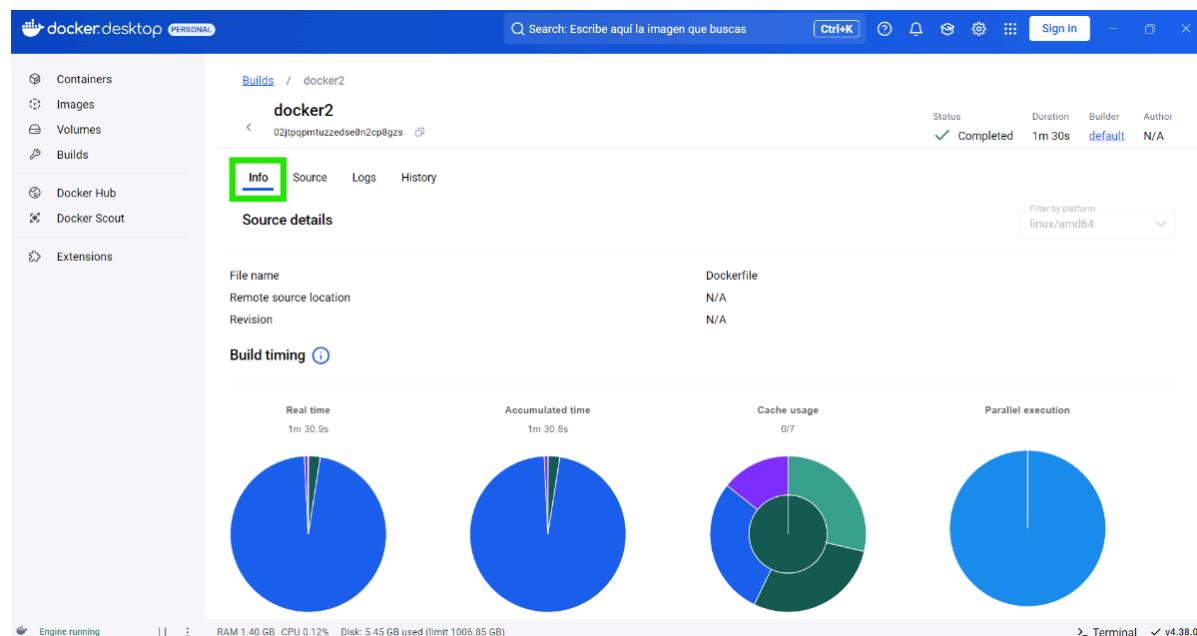
Y ejecutará `python3 script.py` automáticamente.

Una vez explicado de manera muy rápida y sencilla que es un *Dockerfile*, volvamos a qué apartados podemos encontrar dentro de un "Build".

Cuando hacemos click en algún docker build de nuestra lista, nos aparecen diferentes secciones: "**Info", "Source", "Logs" e "History".**



Sección Info:



El apartado **Info** proporciona **información detallada** sobre el proceso de construcción de la imagen que estás realizando. Aquí puedes ver detalles sobre el contexto de construcción, las imágenes involucradas y otros parámetros relacionados.

Funcionalidad:

- **Contexto de construcción:** Muestra el directorio desde donde se está construyendo la imagen. Este contexto contiene el `Dockerfile` y los archivos necesarios que serán incluidos en la imagen.
- **Imágenes base:** Muestra la imagen base que se está utilizando para la construcción. Por ejemplo, si estás utilizando una imagen de Node.js como base, se mostrará aquí.
- **Tamaño de la imagen:** Al final del proceso de construcción, puedes ver el tamaño total de la nueva imagen creada.

Sección Source:

The screenshot shows the Docker Desktop interface. On the left, there's a sidebar with options: Containers, Images, Volumes, Builds, Docker Hub, Docker Scout, and Extensions. The main area is titled "Builds / docker2". Below that is a table with one row for "docker2", showing Status as "Completed", Duration as "1m 30s", Builder as "default", and Author as "N/A". There are tabs for "Info", "Source" (which is selected and highlighted with a green border), "Logs", and "History". A dropdown menu says "Filter by platform linux/amd64". The "Source" tab displays the Dockerfile content:

```

1 FROM ubuntu
2
3 RUN apt update && apt upgrade -y
4
5 RUN apt install apache2 -y
6
7 EXPOSE 80
8
9 CMD [ "apache2ctl", "-D", "foreground" ]

```

Te permite ver el **Dockerfile** que estás utilizando para crear la imagen, junto con los archivos que están siendo copiados al contenedor durante el proceso de construcción.

Sección Logs:

The screenshot shows the Docker Desktop interface with the "Logs" tab selected. The table lists the steps of the build process:

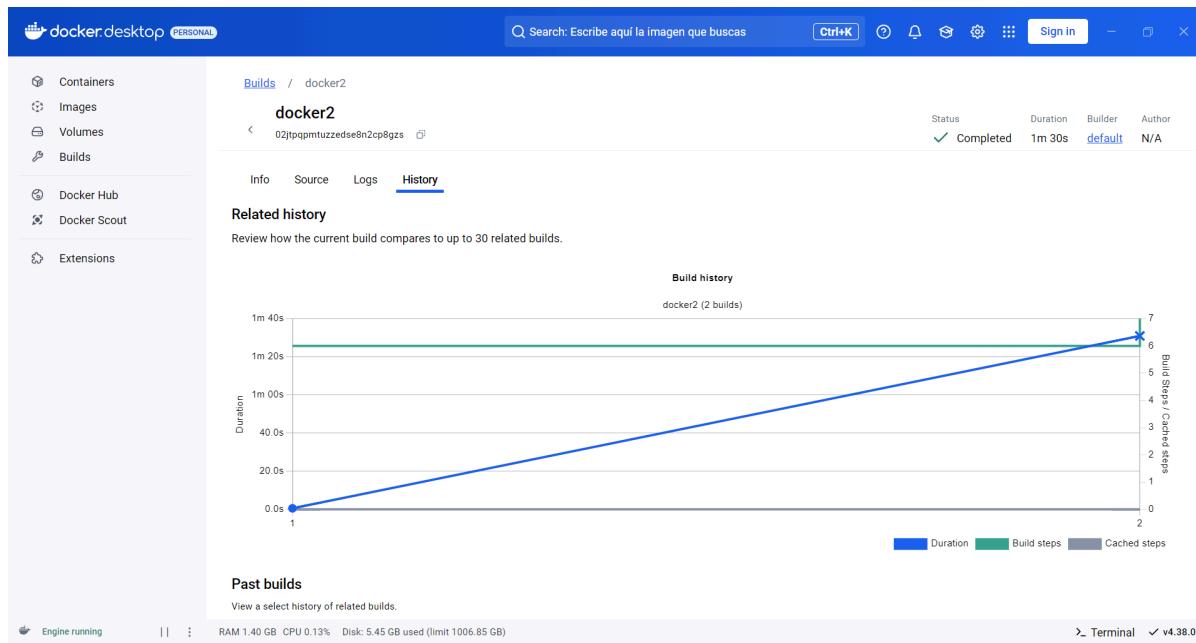
	Message	Duration	Size
✓ INTERNAL	load build definition from Dockerfile transferring	0.0s	0.0s
✓ INTERNAL	load metadata for docker.io/library/ubuntu:latest	2.1s	
✓ INTERNAL	load .dockerignore transferring	0.0s	2 Bytes
✓ 1/3	FROM docker.io/library/ubuntu:latest@sha256:1b8d8ff4777f36f19bfe73ee4df61e3a0b789caeff29caa019539ec7c9a57f95 resolve docker.io/library/ubuntu:latest@sha256:1b8d8ff4777f36f19bfe73ee4df61e3a0b789caeff29caa019539ec7c9a57f95 done done done	0.0s	1.1 kB -1.-7s 424 Bytes -1.-5s 2.2 kB -1.-1s
✓ 2/3	RUN apt update && apt upgrade -y	39.9s	

Muestra los **registros del proceso de construcción**. Aquí puedes ver qué está sucediendo durante el proceso de construcción de la imagen, incluidos los errores, advertencias y mensajes informativos generados por cada paso del **Dockerfile**.

Funcionalidad:

- Ver mensajes detallados:** Cada paso del proceso de construcción es registrado aquí. Si un paso como `RUN npm install` falla, los logs te darán detalles sobre por qué falló.
- Mensajes de error y advertencia:** Si hay algún problema durante la construcción (por ejemplo, una dependencia faltante o un comando erróneo), se mostrará en los logs.

Sección History:



Te muestra el **historial de capas** de la imagen que estás construyendo. En Docker, las imágenes se construyen a partir de **capas** que corresponden a cada instrucción en el `Dockerfile` (por ejemplo, `FROM`, `RUN`, `COPY`). El historial te permite ver el detalle de todas las capas creadas durante la construcción de la imagen.

Funcionalidad:

- **Visualizar capas de la imagen:** Cada paso de tu `Dockerfile` genera una nueva capa en la imagen. Aquí podrás ver estas capas y cómo cada una contribuye al tamaño final de la imagen.
- **Detalles sobre cada capa:** Puedes ver detalles sobre cada capa, como el comando que se ejecutó para crearla, su tamaño y cuándo fue creada.
- **Optimización de imágenes:** El historial te permite ver cómo se construye la imagen y si hay pasos innecesarios o redundantes que pueden ser optimizados.

3.6. Networks (Redes)

Docker permite crear **redes virtuales** para que los contenedores se comuniquen entre sí. Desde aquí puedes:

- **Gestionar redes:** Crear, ver y eliminar redes personalizadas, como redes de puente (`bridge`) o redes de contenedor a contenedor.

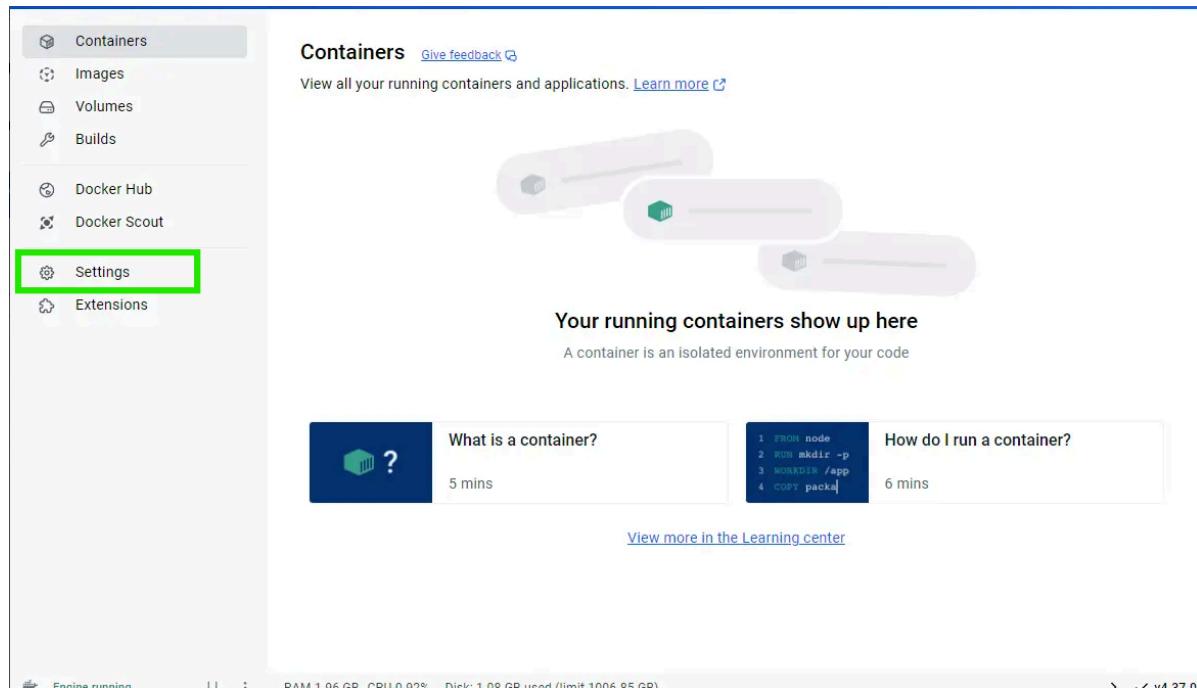
3.7. Extensions (Extensões)

Las **extensiones** son aplicaciones y herramientas adicionales que puedes instalar dentro de Docker Desktop para mejorar o ampliar sus funcionalidades. Algunas extensiones son proporcionadas por Docker, mientras que otras provienen de la comunidad o de desarrolladores externos.

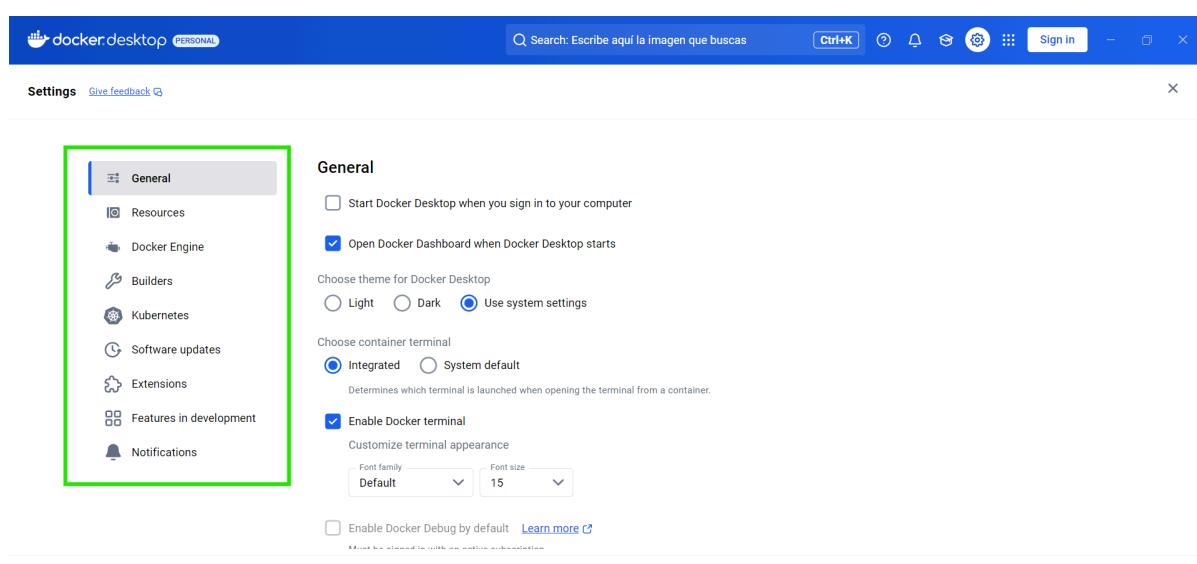
Estas extensiones pueden ayudarte a realizar **tareas específicas**, como:

- Monitorización de contenedores y aplicaciones.
- Gestión avanzada de redes y volúmenes.
- Integración con servicios de nube.
- Herramientas para desarrollo y pruebas.

3.8. Settings (Configuración)



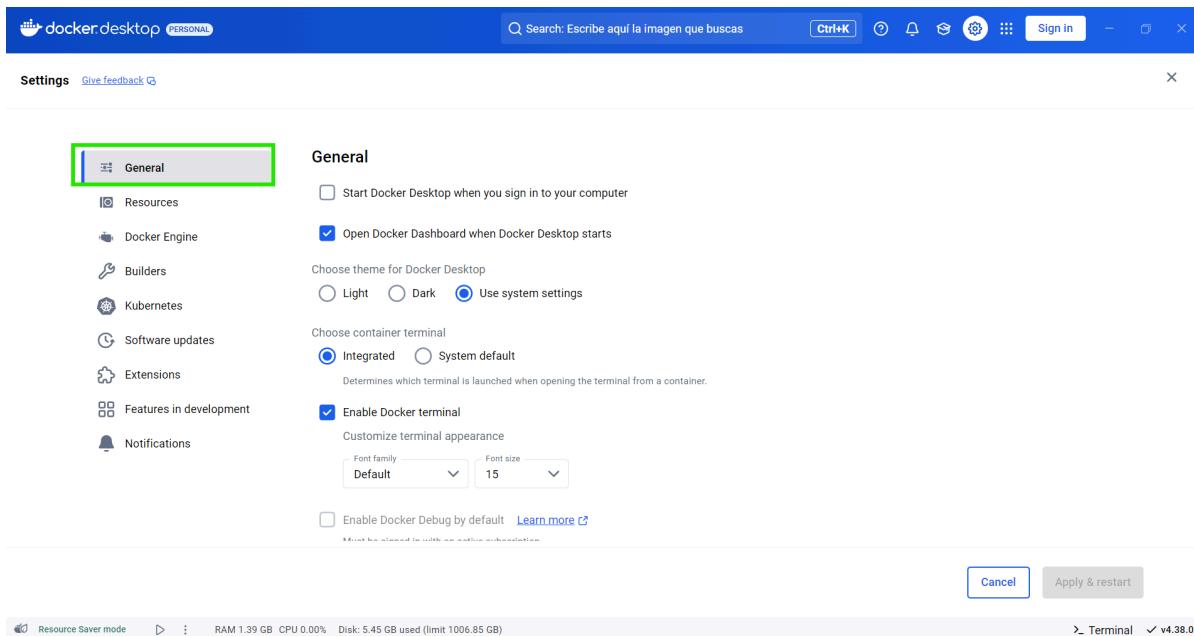
Aquí puedes ajustar varias configuraciones de Docker Desktop, tales como:



3.8.1.General

En **Docker Desktop** dentro de **Settings > General**, puedes configurar varias opciones clave para el comportamiento general de Docker. A continuación, os muestro un pequeño resumen de las más importantes:

- **Open Docker Dashboard when Docker Desktop starts** → Configura si el Dashboard se abre automáticamente al iniciar Docker Desktop.
- **Choose container terminal** → Selecciona qué terminal se usará al abrir un contenedor.
 - **Integrated** → Usa el terminal integrado en Docker Desktop.
 - **System default** → Usa el terminal del sistema (cmd, PowerShell, etc.).
- **Enable Docker terminal** → Activa el terminal de Docker dentro de la aplicación.
- **Customize terminal appearance** → Personaliza la apariencia del terminal:
 - **Font family** → Tipo de fuente.
 - **Font size** → Tamaño de fuente (predeterminado: 15).
- **Enable Docker Debug by default** → Activa el modo de depuración de Docker por defecto.
- **Expose daemon on tcp://localhost:2375 without TLS** → Expone el daemon de Docker sin TLS (⚠ riesgo de seguridad).
- **Use the WSL 2 based engine (Windows only)** → Usa WSL 2 en lugar de Hyper-V en Windows.
- **Add the *.docker.internal names to the host's /etc/hosts file** → Permite resolver los nombres de dominio **.docker.internal** desde el host y los contenedores.
- **Use containerd for pulling and storing images** → Usa **containerd** en lugar de Docker Engine para manejar imágenes.
- **Send usage statistics** → Envía estadísticas de uso a Docker.
- **Use Enhanced Container Isolation** → Mejora la seguridad para prevenir que los contenedores accedan a la VM de Linux (requiere suscripción Business).
- **Show CLI hints** → Muestra sugerencias en la CLI al ejecutar comandos Docker.
- **Enable Scout image analysis** → Activa el análisis de imágenes con **Docker Scout**.
- **Enable background Scout SBOM indexing** → Inicia automáticamente el indexado **SBOM** de imágenes cuando se crean o inspeccionan.



1. Advanced:

Configura opciones avanzadas como la cantidad de **CPU**, **memoria RAM** y **swap** para optimizar el rendimiento de Docker según las necesidades específicas de tu máquina y los proyectos.

2. Proxies:

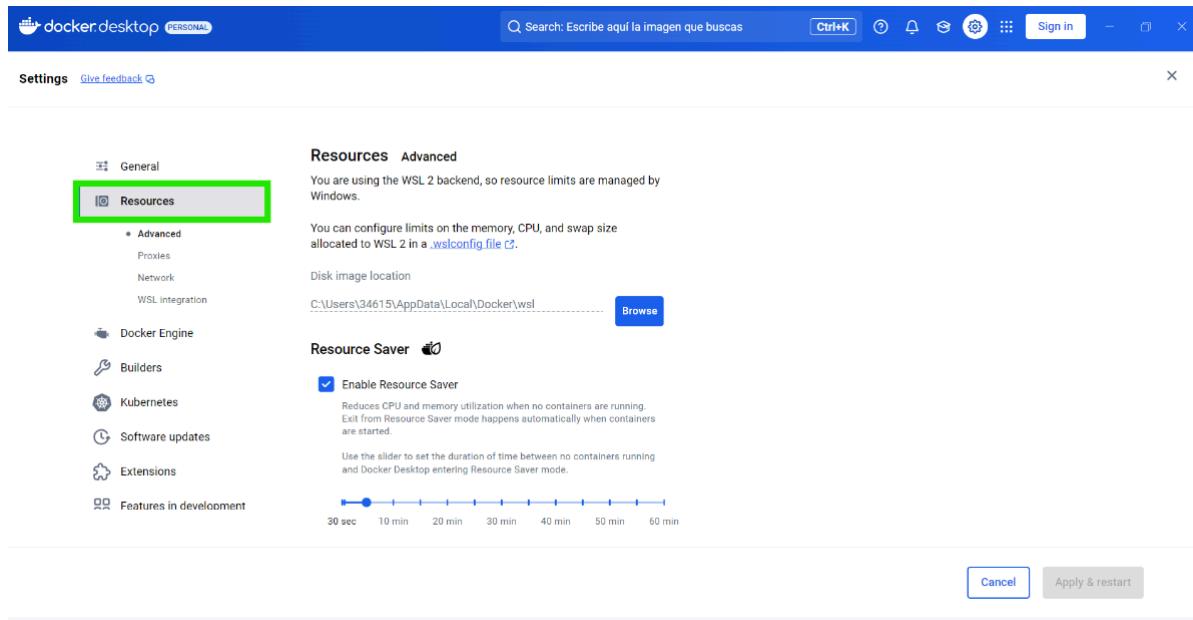
Aquí puedes configurar un **proxy** si tu red requiere uno para acceder a internet. Esto es común en entornos corporativos o redes restringidas, donde Docker necesita acceder a recursos externos como **Docker Hub** a través de un proxy.

3. Network:

Gestiona las opciones de red avanzadas. Aquí puedes configurar aspectos como el **DNS** para los contenedores y cómo Docker maneja las redes entre contenedores y entre el host y los contenedores.

4. WSL Integration:

Esta opción está disponible para usuarios de **Windows** y permite configurar la **integración con WSL 2** (Windows Subsystem for Linux 2). Puedes habilitar o deshabilitar la integración de Docker con distribuciones de Linux instaladas en WSL 2 y seleccionar qué distribuciones pueden usar Docker para ejecutar contenedores.



¿Para qué sirve?

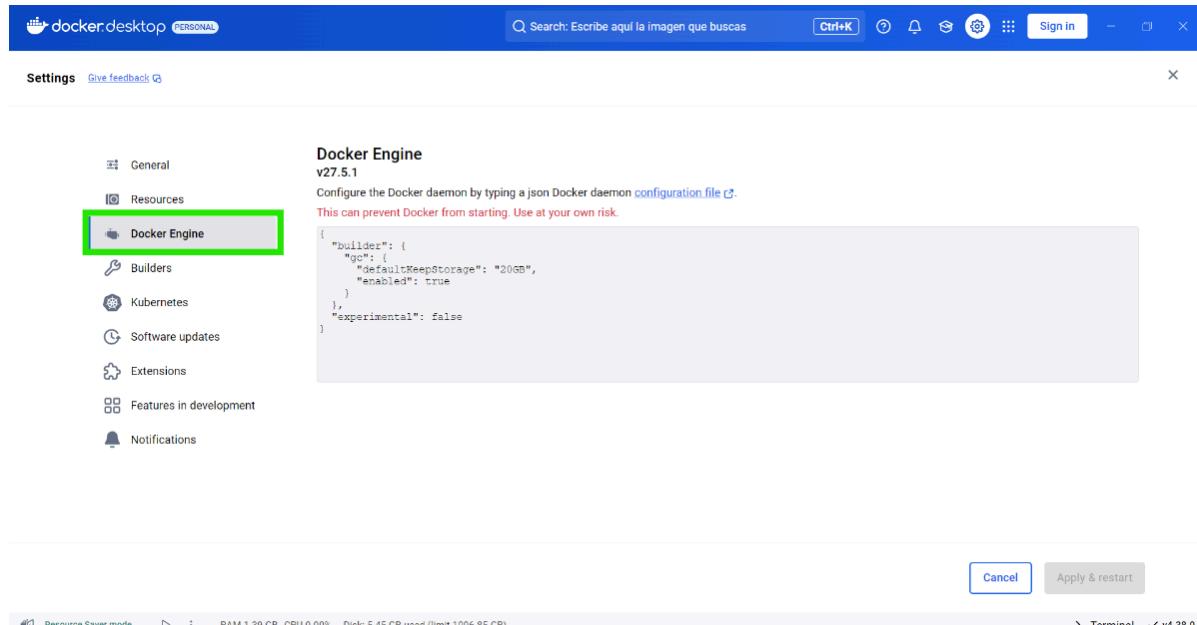
En la sección **Resources**, puedes configurar la cantidad de recursos del sistema que Docker usará, como la CPU, la memoria RAM y el espacio en disco. Esto es especialmente importante cuando trabajas con contenedores que requieren más recursos o si tienes un sistema con capacidades limitadas.

Funcionalidad:

- **CPU:** Puedes asignar el número de **núcleos de CPU** que Docker puede utilizar. Si tienes un sistema con múltiples núcleos, puedes asignar más para mejorar el rendimiento de los contenedores, especialmente si estás ejecutando aplicaciones pesadas.
- **Memoria (RAM):** Permite configurar la cantidad de **memoria RAM** que Docker puede utilizar. Si tus contenedores están experimentando problemas de rendimiento por falta de memoria, puedes aumentar esta cantidad.
- **Swap:** Docker usa **swap** (memoria virtual) cuando la memoria RAM está llena. Puedes ajustar el tamaño del swap para mejorar el manejo de memoria.

- **Disco:** Configura el espacio en disco que Docker puede usar para almacenar imágenes, contenedores y otros datos. Si estás trabajando con muchos contenedores o imágenes grandes, puede que necesites ajustar este valor.
- **Red:** Docker te permite configurar cómo usa la red en tu sistema. Aquí puedes ajustar aspectos como el **puerto de Docker**, la configuración de redes, etc.

3.8.3. Docker Engine



¿Para qué sirve?

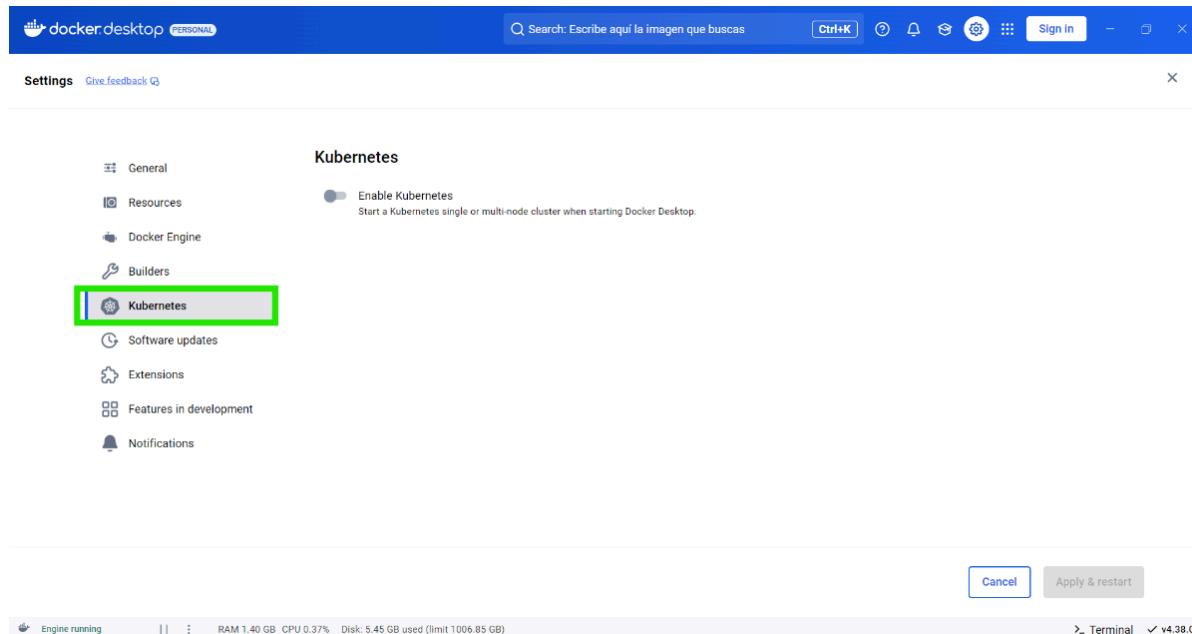
En la sección **Docker Engine**, puedes configurar directamente los parámetros del motor Docker utilizando un archivo de configuración JSON. Esto te permite personalizar aún más la forma en que Docker opera en tu máquina.

Funcionalidad:

- **Configuración avanzada:** Puedes editar la configuración del motor Docker directamente, modificando parámetros como la cantidad de memoria compartida, las opciones de registro o la dirección de escucha del *Docker Daemon*.*
- **Contenedores predeterminados:** Puedes personalizar las opciones relacionadas con cómo se deben crear o ejecutar los contenedores.
- **Inicio de Docker:** Puedes configurar qué tipo de comportamiento debe tener Docker al iniciar (por ejemplo, si debe intentar reiniciar los contenedores automáticamente).

*Un Docker Daemon, (dockerd) es el proceso en segundo plano que gestiona todas las operaciones de Docker en el sistema. Se encarga de crear, ejecutar y supervisar contenedores, gestionar imágenes, manejar redes y volúmenes, y comunicarse con repositorios de imágenes. También expone una API para interactuar con el **Docker Client**, que es el que envía las instrucciones al daemon. En resumen, el Docker Daemon es el "motor" que ejecuta y coordina todas las acciones relacionadas con contenedores en nuestra máquina.

3.8.4. Kubernetes



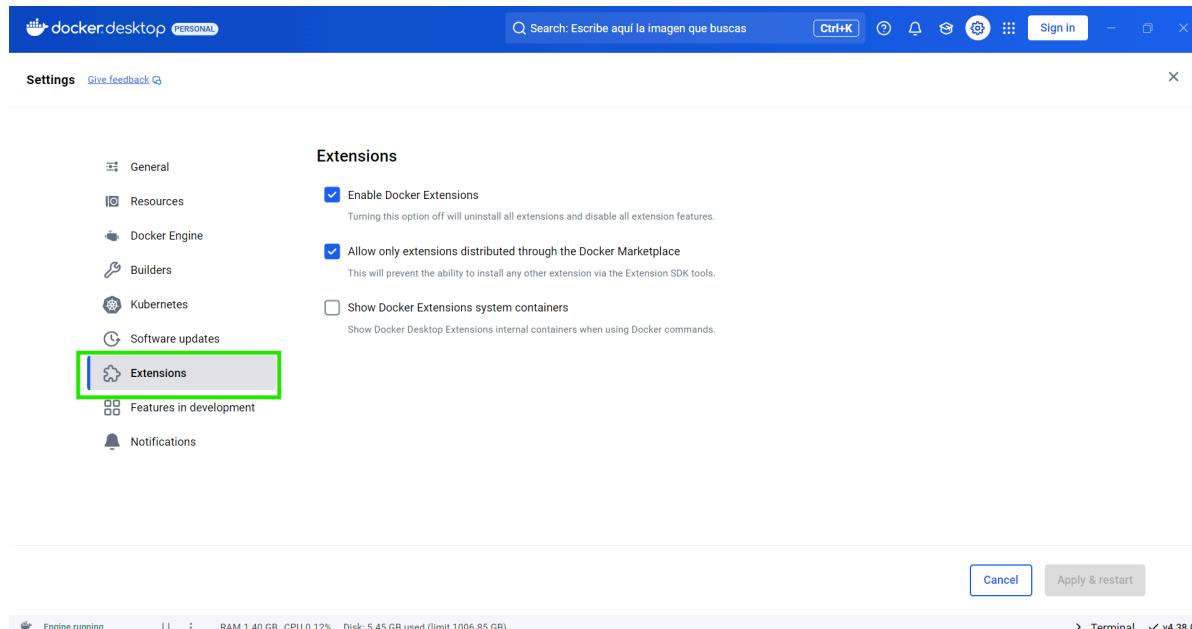
¿Para qué sirve?

Docker Desktop viene con la opción de integrar **Kubernetes** (un sistema de orquestación de contenedores) directamente. En esta sección, puedes configurar y gestionar Kubernetes dentro de Docker Desktop.

Funcionalidad:

- **Habilitar o deshabilitar Kubernetes:** Si no necesitas Kubernetes en tu proyecto, puedes desactivarlo desde esta sección. Si trabajas con aplicaciones de **microservicios** o en un **entorno de producción**, Kubernetes puede ser útil para la orquestación de contenedores.
- **Configurar el clúster:** Si decides habilitar Kubernetes, Docker Desktop te permite **configurar tu clúster Kubernetes** localmente para pruebas y desarrollo.
- **Borrar el clúster:** Si ya no necesitas el clúster de Kubernetes, puedes eliminarlo y liberar los recursos que estaba utilizando.

3.8.5. Extensions (Extensiones)



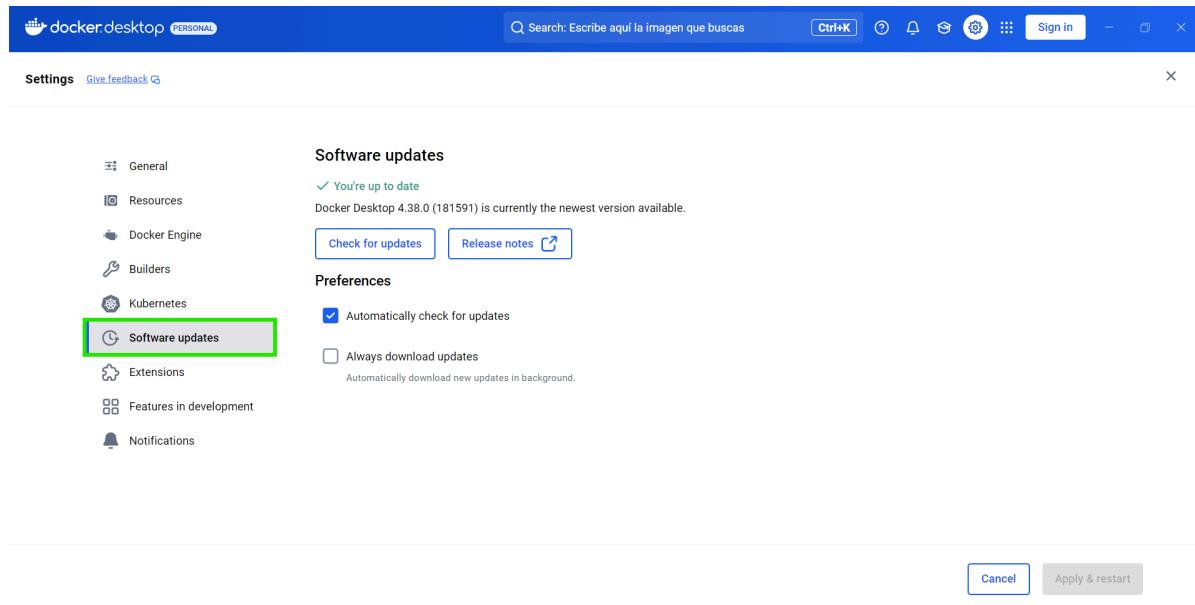
¿Para qué sirve?

Esta sección te permite gestionar las **extensiones** que puedes instalar para ampliar las funcionalidades de Docker Desktop.

Funcionalidad:

- **Explorar e instalar extensiones:** Puedes ver qué extensiones están disponibles, instalarlas y configurarlas para ampliar las funcionalidades de Docker Desktop. Algunas de las extensiones populares incluyen herramientas para monitoreo, integración con Kubernetes y servicios en la nube, entre otras.
- **Gestionar las extensiones:** Puedes ver las extensiones instaladas, actualizarlas o eliminarlas según tus necesidades.

3.8.6. Updates (Actualizaciones)



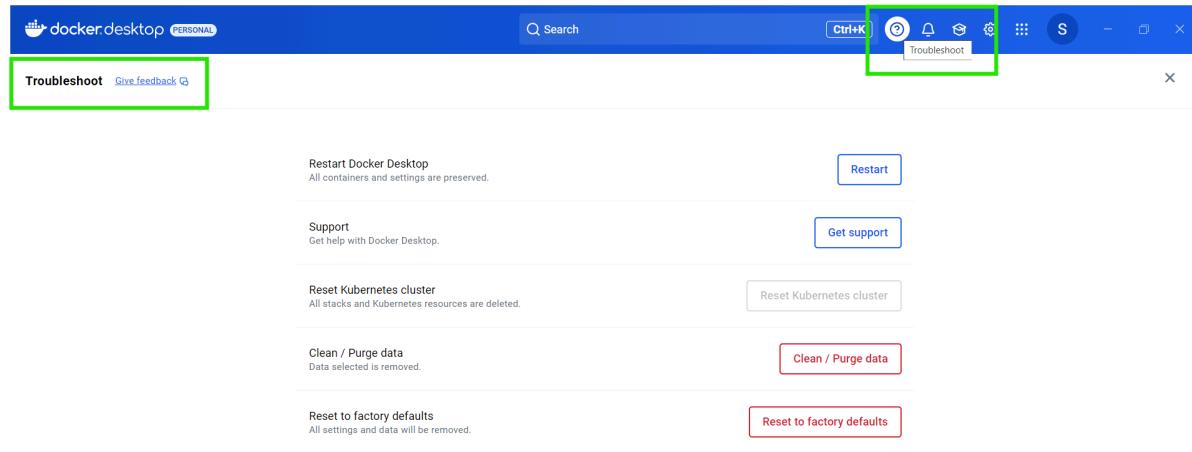
¿Para qué sirve?

Esta sección nos permite gestionar las **actualizaciones** de Docker Desktop.

Funcionalidad:

- **Verificar actualizaciones:** Docker Desktop verifica automáticamente si hay actualizaciones disponibles y te notifica si tienes una nueva versión.
- **Configurar actualizaciones automáticas:** Puedes activar o desactivar las actualizaciones automáticas de Docker, lo que es útil si prefieres controlar cuándo realizar las actualizaciones.
- **Historial de actualizaciones:** Puedes ver un historial de las versiones de Docker que has instalado previamente.

3.8.7. Troubleshoot (Solucionar problemas)



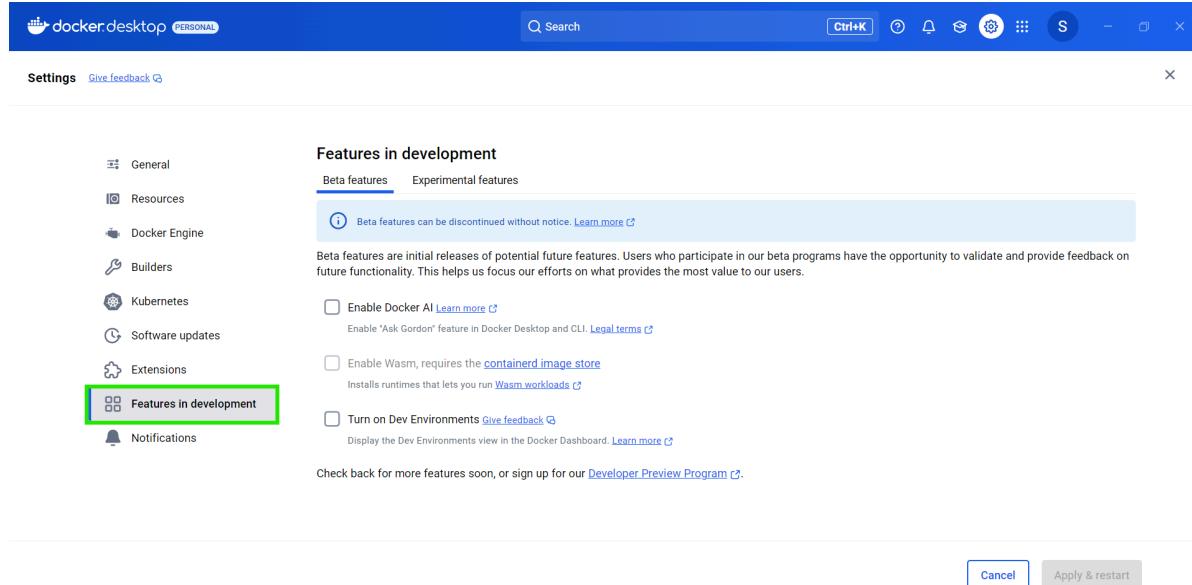
¿Para qué sirve?

La sección **Troubleshoot** nos ayuda a diagnosticar problemas y errores en Docker Desktop.

Funcionalidad:

- Reiniciar Docker**: Si Docker no está funcionando correctamente, puedes reiniciarlo desde esta sección para solucionar problemas temporales.
- Diagnóstico**: Docker Desktop incluye una herramienta de diagnóstico que analiza el estado de la aplicación y te proporciona información útil para resolver problemas.
- Reiniciar configuraciones**: Si Docker está experimentando problemas graves, puedes **restaurar los ajustes de fábrica** o **borrar datos** para empezar de nuevo sin perder configuraciones importantes.

3.8.8. Experimental Features (Funciones Experimentales)



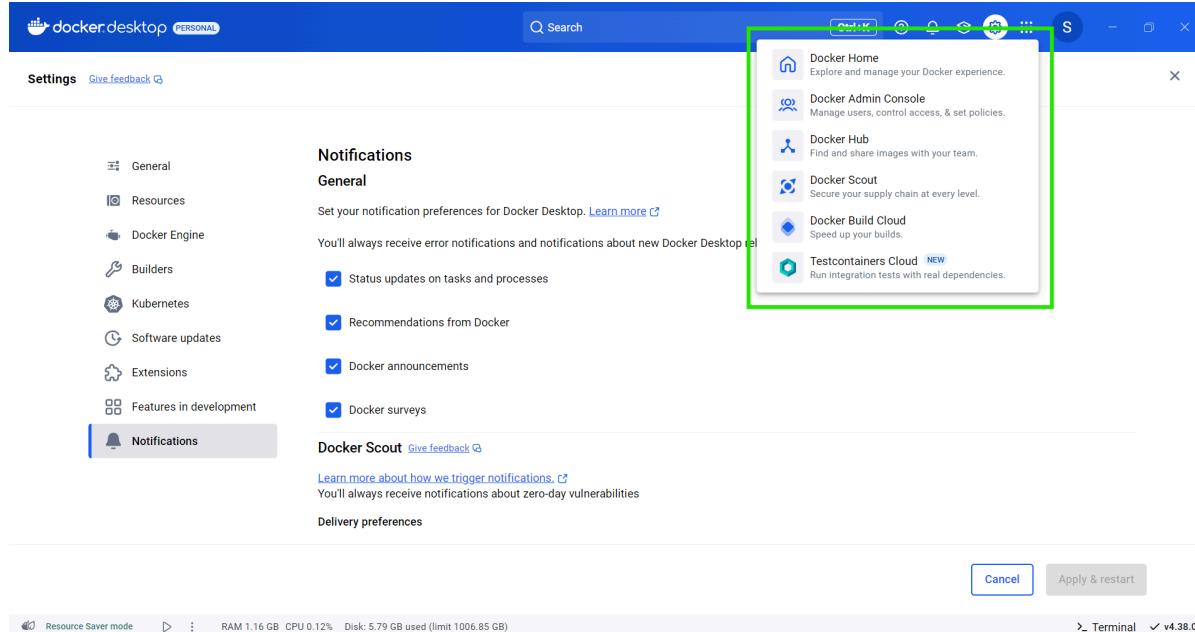
¿Para qué sirve?

En esta sección, puedes activar o desactivar las **funciones experimentales** de Docker.

Funcionalidad:

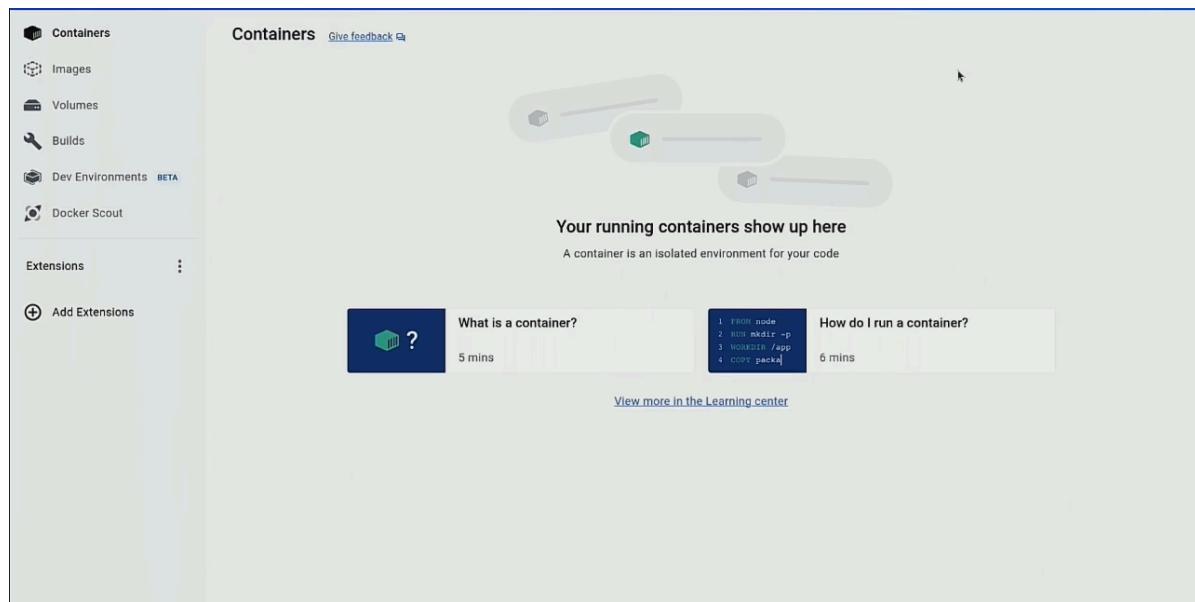
- **Activar características beta:** Algunas funciones de Docker están en versión beta o experimental. Si activas esta opción, puedes probar características nuevas que aún no están completamente disponibles para todos los usuarios.
- **Personalización avanzada:** Las características experimentales pueden proporcionar nuevas capacidades, pero no siempre están completamente pulidas.

4. Otras áreas y secciones:



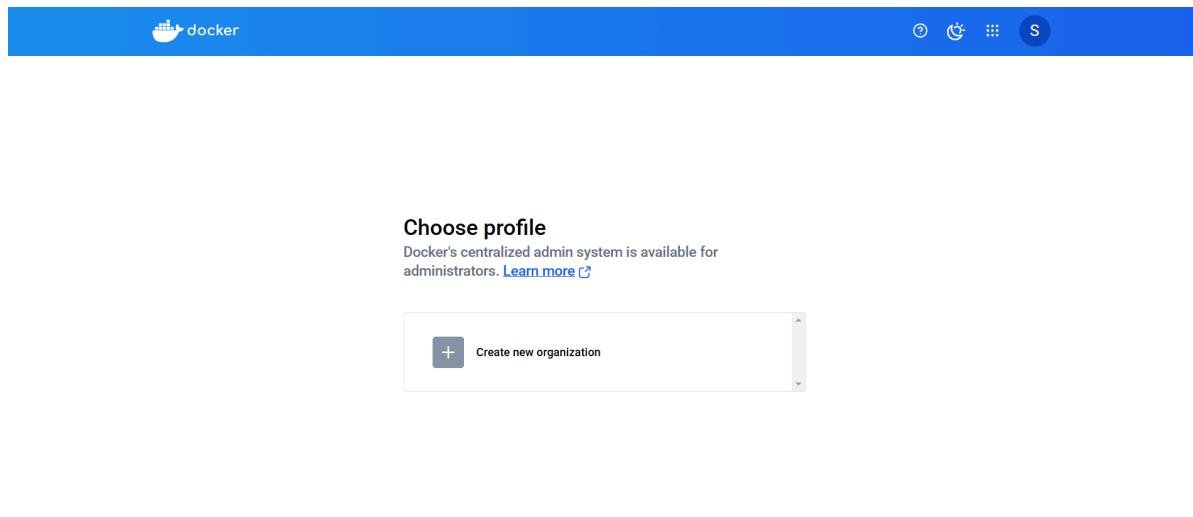
4.1. Docker Home:

Es la pantalla principal de Docker Desktop, donde podemos ver el estado general de Docker, gestionar contenedores, imágenes y redes, así como acceder a las configuraciones y herramientas.



4.2. Docker Admin Console:

Un panel administrativo para gestionar configuraciones avanzadas, usuarios y recursos de Docker en tu máquina o entorno de trabajo.



4.3. Docker Hub:

La plataforma de almacenamiento y distribución de imágenes Docker, donde podemos buscar, descargar o subir imágenes oficiales o personalizadas.

A screenshot of the Docker Hub homepage. The top navigation bar includes links for 'Explore', 'Repositories' (which is underlined), 'Organizations', and 'Usage'. There's also a search bar with placeholder text 'Search Docker Hub' and a 'ctrl+K' keyboard shortcut. On the right are various user icons. The main content area has a blue background with the text 'Welcome to Docker' and 'Download the desktop application'. It features a 'Download for Windows' button and notes that it's also available for Mac and Linux. Below this are three cards: 'Create a Repository', 'Docker Hub Basics', and 'Language-Specific Guides'.

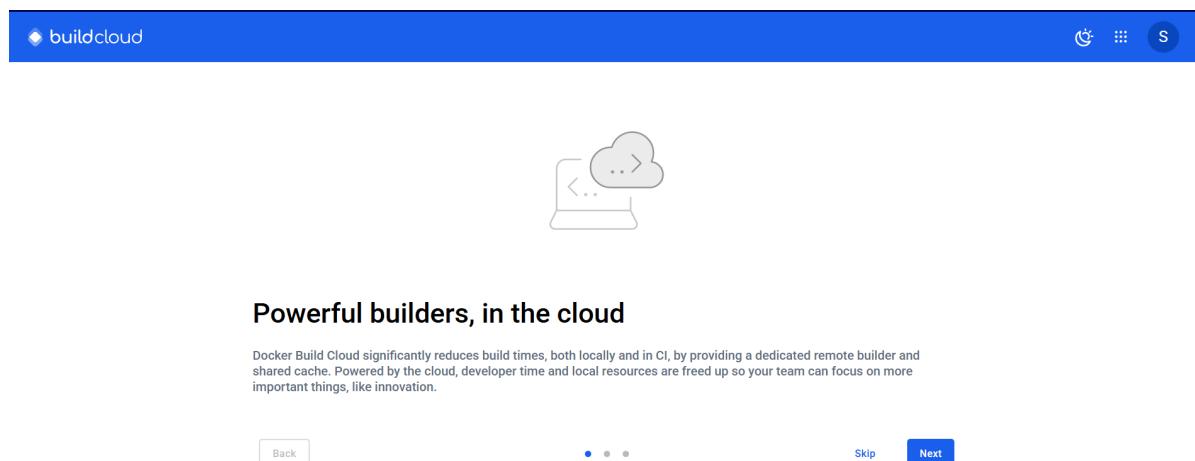
4.4. Docker Scout:

Herramienta que analiza imágenes Docker en busca de vulnerabilidades, dependencias y otros problemas de seguridad.

A screenshot of the Docker Scout landing page. The top navigation bar has the Docker Scout logo and icons. The main visual is a comparison between Docker Desktop and Docker Scout, showing how Docker Scout provides more detailed reports. Below this, the text 'Unlock the power of Docker Scout across your ecosystem' is displayed, along with a note about getting detailed reports on image vulnerabilities and remediation suggestions. At the bottom, there are navigation buttons for 'Skip' and 'Next'.

4.5. Docker Build Cloud:

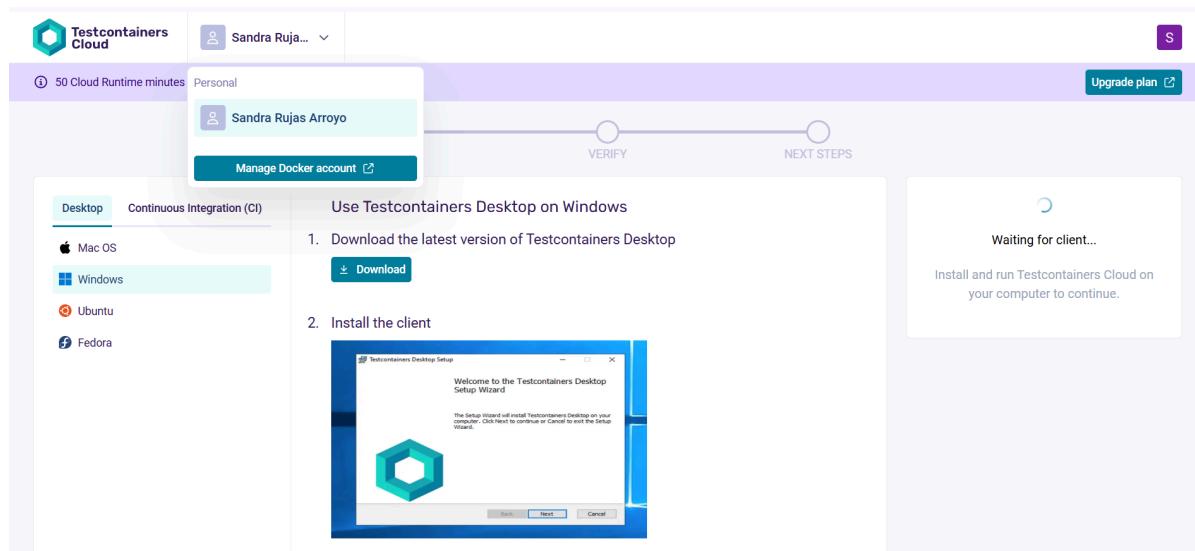
Funcionalidad que permite construir imágenes Docker en la nube, optimizando recursos y mejorando el rendimiento de la construcción de contenedores.



The screenshot shows the Docker Build Cloud landing page. At the top, there's a blue header with the 'buildcloud' logo and some icons. Below the header is a large cloud icon with arrows pointing in and out. The main title 'Powerful builders, in the cloud' is centered. A subtitle explains that Docker Build Cloud significantly reduces build times by providing a dedicated remote builder and shared cache. Below the text are navigation buttons: 'Back', a progress bar with three dots, 'Skip', and a 'Next' button.

4.6. Testcontainers Cloud:

Servicio que permite ejecutar pruebas automatizadas de contenedores en la nube, especialmente útil para pruebas de integración con contenedores de bases de datos u otros servicios.



The screenshot shows the Testcontainers Cloud dashboard. It features a sidebar with account information ('Sandra Ruja...'), a purple header bar with '50 Cloud Runtime minutes' and an 'Upgrade plan' button, and a main area divided into sections: 'Desktop' (selected), 'Continuous Integration (CI)', and 'Use Testcontainers Desktop on Windows'. The 'Desktop' section lists supported operating systems: Mac OS, Windows, Ubuntu, and Fedora. It includes steps for download and client installation, with a screenshot of the 'Testcontainers Desktop Setup' wizard. To the right, a box says 'Waiting for client...' with instructions to install and run the client.

5. Resumen final de las funcionalidades clave de Docker Desktop

5.1. Descargar e instalar Docker Desktop

- En **Windows o macOS**, podemos descargar Docker Desktop desde el sitio web oficial de Docker, tan sólo debemos seguir los pasos del asistente de instalación que nos proporcionan.

5.2. Usar Docker Desktop

Una vez que Docker Desktop esté instalado y en funcionamiento:

1. **Iniciamos Docker Desktop** desde el ícono en la barra de tareas o en el dock (en macOS).
2. **Accedemos a las áreas de "Images", "Containers" y "Settings".**

3. **Gestionamos contenedores, imágenes, volúmenes y redes** de manera visual, sin necesidad de usar comandos en la terminal.

5.3. Configuración de recursos

Deberemos asegurarnos de que Docker tenga suficientes recursos (CPU, RAM) para ejecutar nuestras aplicaciones. Puedes ajustarlo desde **Settings > Resources** (*sección explicada en el punto 3.8.2. Resources (Recursos)*).

5.4. Crear y ejecutar contenedores

- **Crear contenedores:** Con las imágenes descargadas, podemos crear contenedores para ejecutar aplicaciones o servicios.
- **Ejecutar contenedores:** Desde Docker Desktop, podemos iniciar contenedores directamente, sin necesidad de usar la línea de comandos, aunque la terminal sigue estando disponible para mayor flexibilidad.

5.5. Docker Compose

Docker Compose es una herramienta que permite definir y ejecutar aplicaciones con múltiples contenedores, generalmente usados para aplicaciones complejas. Podemos gestionar nuestras aplicaciones multi-contenedor usando archivos `docker-compose.yml` que describen los servicios.

5.6. Terminal integrada

Aunque la GUI de Docker Desktop es muy intuitiva, Docker también incluye una terminal integrada donde podemos ejecutar todos los comandos de Docker tradicionales:

- **docker run:** Crea un contenedor y lo ejecuta.
- **docker ps:** Muestra los contenedores activos.
- **docker build:** Construye imágenes a partir de un Dockerfile.
- **docker-compose up:** Levanta todos los servicios definidos en un archivo `docker-compose.yml`.

5.7. Integración con Docker Hub

Docker Desktop se integra con **Docker Hub** (el registro público de imágenes de Docker). Desde la GUI puedes:

- **Buscar imágenes** en Docker Hub.
- **Subir imágenes** que crees localmente para compartirlas o reutilizarlas en otros entornos.

5.8. Docker Desktop en WSL 2 (Windows)

En sistemas Windows, Docker Desktop usa **WSL 2** (Windows Subsystem for Linux) para ejecutar contenedores de manera eficiente. Esto mejora el rendimiento y permite una integración más fluida entre Windows y Linux. WSL 2 es especialmente útil para trabajar con aplicaciones basadas en Linux desde una máquina Windows.

En resumen, Docker Desktop ofrece una plataforma poderosa y fácil de usar para desarrollar, gestionar y ejecutar contenedores en nuestra máquina local. Con herramientas como **Docker Home** para gestionar tus contenedores e imágenes, **Docker Hub** para acceder a un vasto repositorio de imágenes, y **Docker Scout** para mejorar la seguridad de tus proyectos, Docker Desktop facilita el flujo de trabajo de desarrollo y prueba. Además, con funcionalidades avanzadas como **Docker Build Cloud** y **Testcontainers Cloud**, puedes optimizar la construcción y prueba de contenedores en la nube. Docker Desktop no solo simplifica la administración de contenedores, sino que también nos brinda un entorno flexible y escalable para nuestros proyectos, ya sea en desarrollo local o en producción.

EJERCICIO 2 - SERVIDOR DE BASE DE DATOS

Realizado por Andrea Gómez Fueyo y Sandra Rujas

EJERCICIO 2 - SERVIDOR DE BASE DE DATOS

1. Descarga de imagen.
2. Despliegue de contenedor.
3. Acceso a la base de datos.
4. Borrado del contenedor.
5. Comprobación de la existencia de datos.
6. Creación de nuevo contenedor.
7. Comprobación de la existencia de la tabla en la base de datos.
8. Eliminación de imagen.
9. Eliminación de volumen, imagen y contenedor.

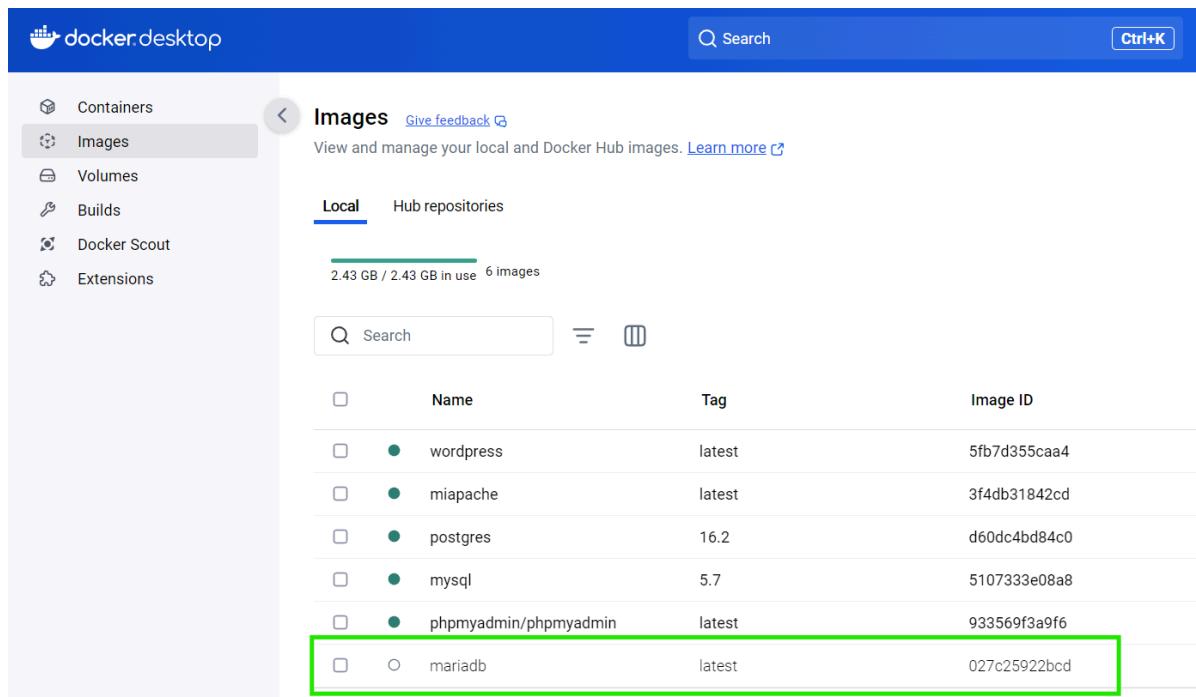
1. Descarga de imagen.

Abre Docker Desktop. Busca mariadb en la sección de imágenes. Selecciona la imagen oficial. Descárgala si no la tienes.

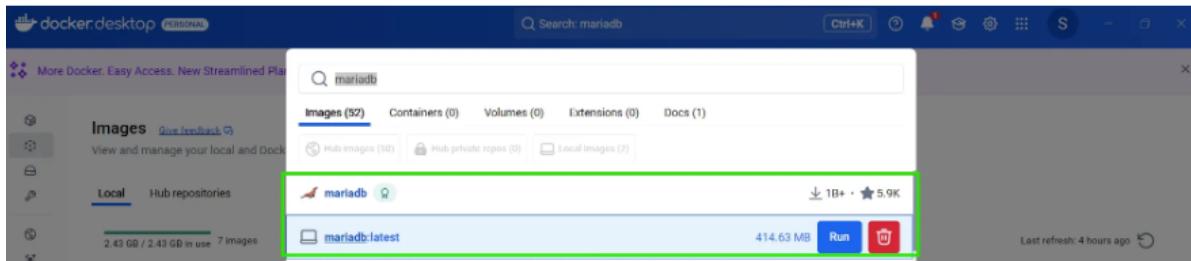
- Si no puedes iniciar sesión, la imagen puedes descargarla mediante comandos.

```
$docker pull mariadb:latest
```

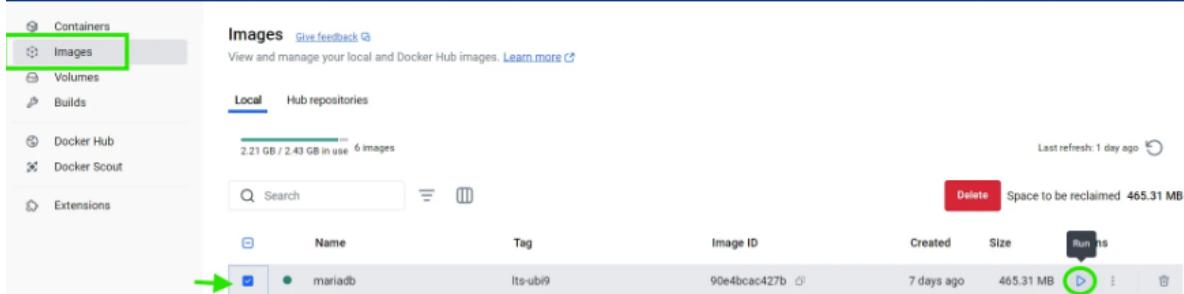
```
C:\Users\34615>docker pull mariadb:latest
latest: Pulling from library/mariadb
5a7813e071bf: Downloading [=====>] 25.22MB/29.75MB
f67c6fbcc0ef5: Download complete
1f731489858b: Download complete
760f6e3db6bf: Download complete
65dd09f27c61: Waiting
2cbd49ab14b1: Downloading [=====>] 11.84MB/89.76MB
640331c2cc76: Waiting
edb426f4a1af: Waiting
```



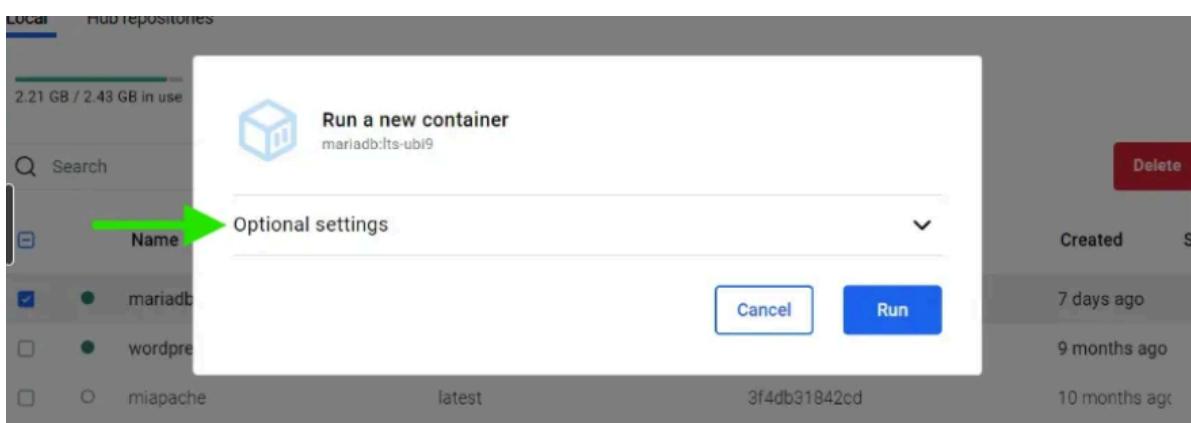
- Si puedes acceder a la búsqueda de imagen, entras en “Images”, buscas la última de mariadb y pulsas pull para descargarla.

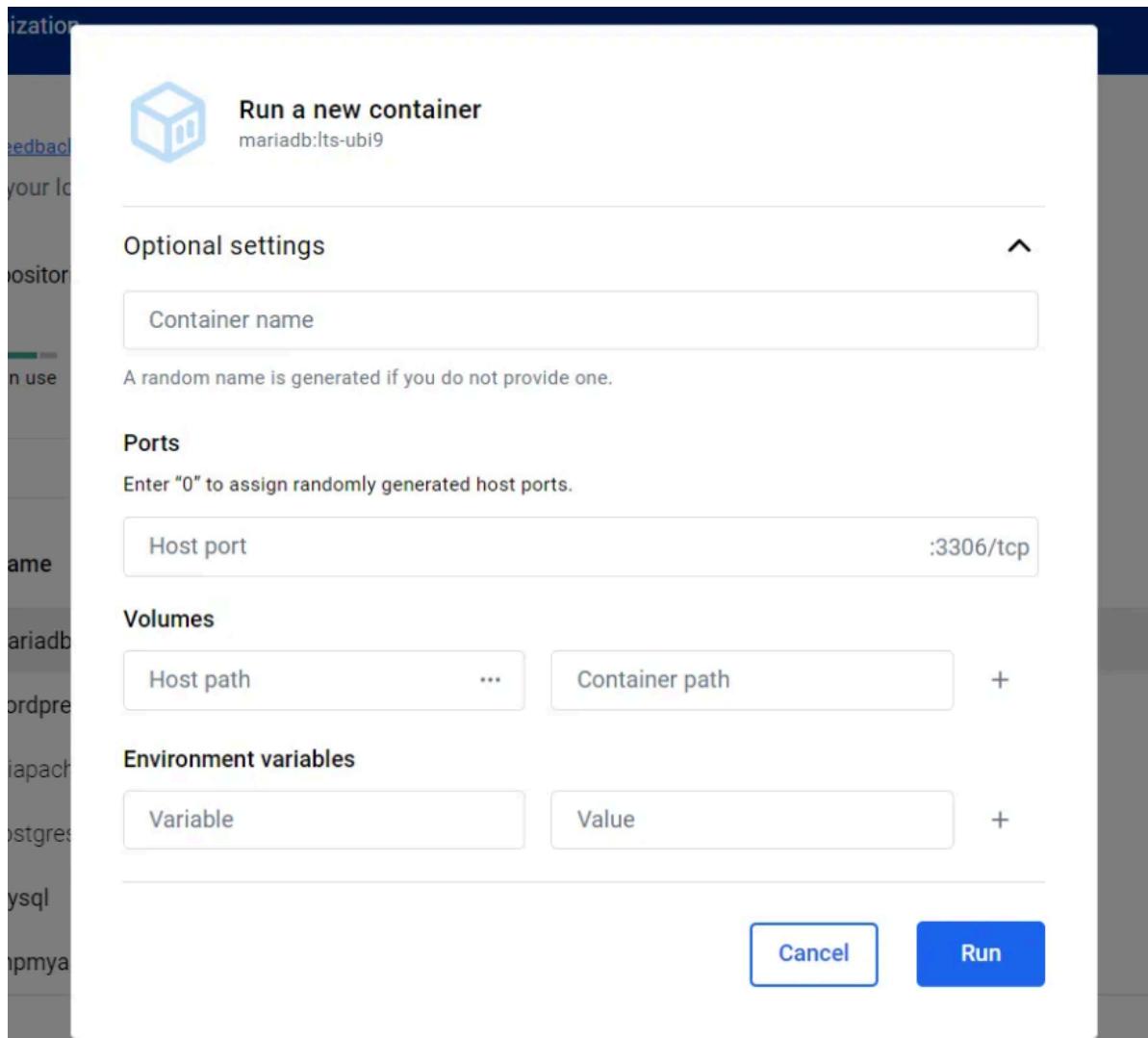


- En la sección de imágenes, seleccionamos la que queremos utilizar, que en nuestro caso es la de mariadb, y pulsamos el botón de run:



- Al darle al run, nos permite crear un nuevo contenedor:

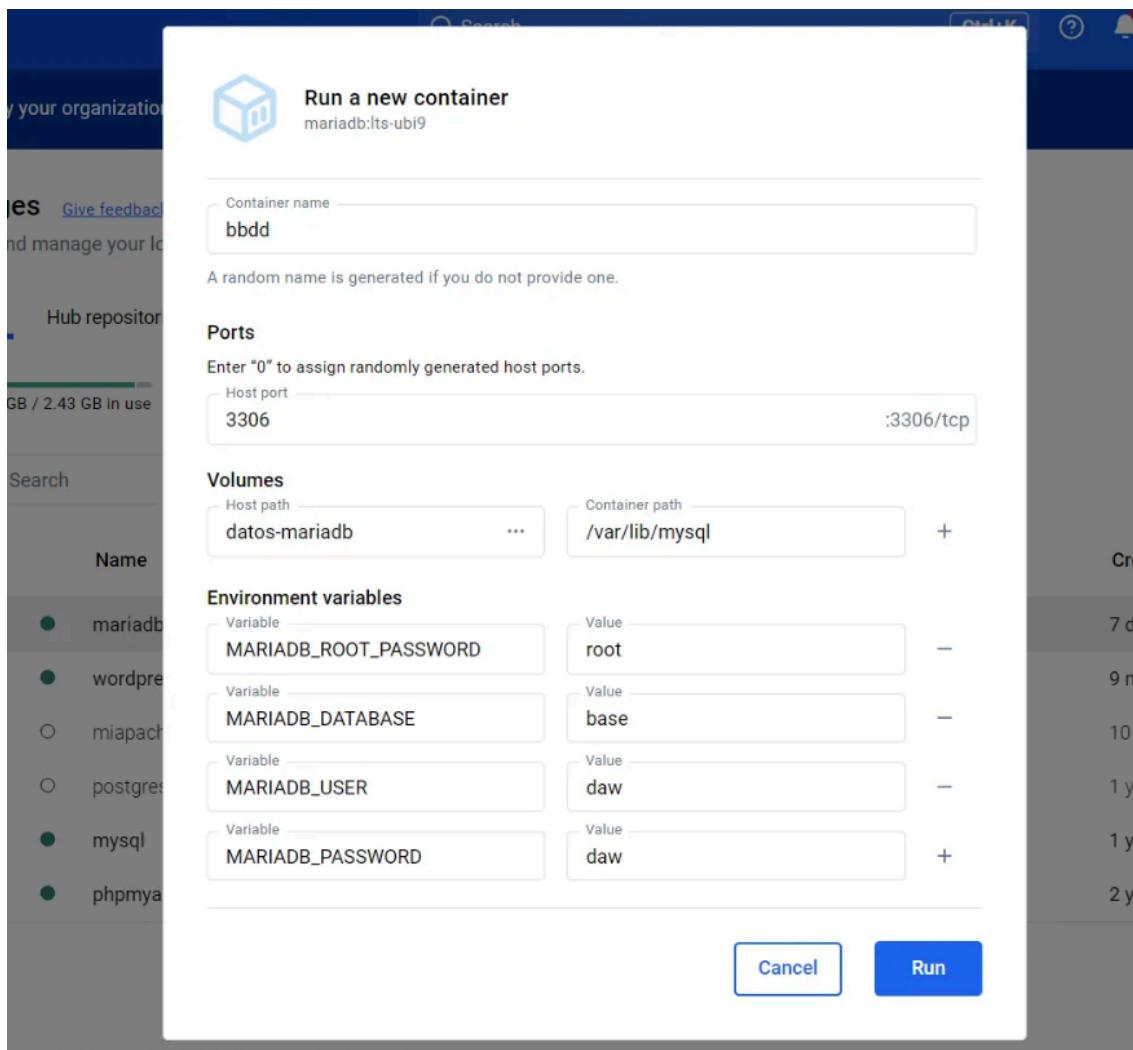




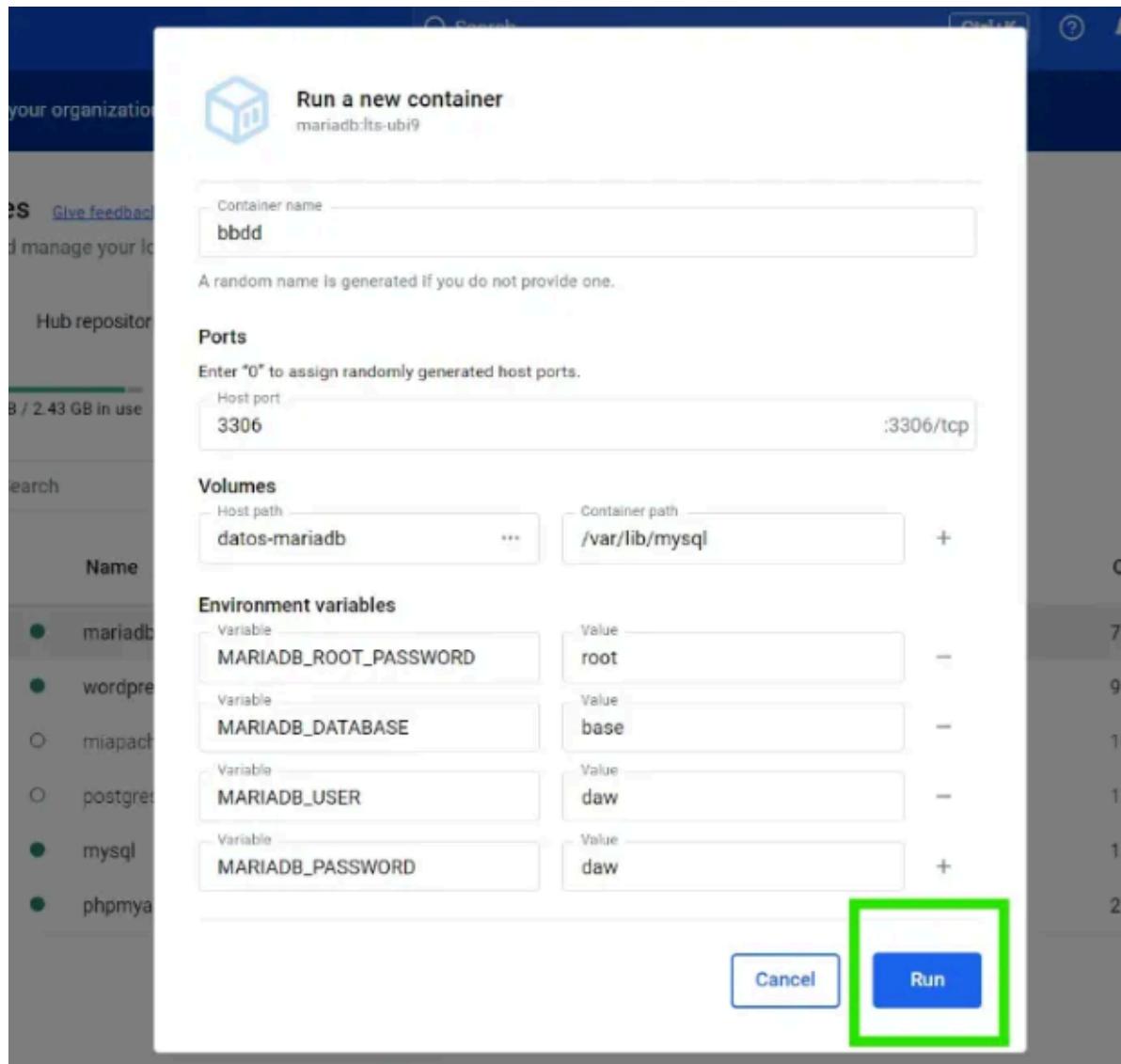
2. Despliegue de contenedor.

Al cual le tenemos que poner las siguientes **características**:

- **Nombre del contenedor:** bbdd .
- **Puerto:** 3306 - debe poder conectarse externamente
- Utiliza un **volumen** llamado datos-mariadb .
- Usa las **variables de entorno** necesarias para que el usuario root tenga la password root , la base de datos por defecto sea base , y se cree un usuario daw , con password daw.



- Arrancamos el contenedor:



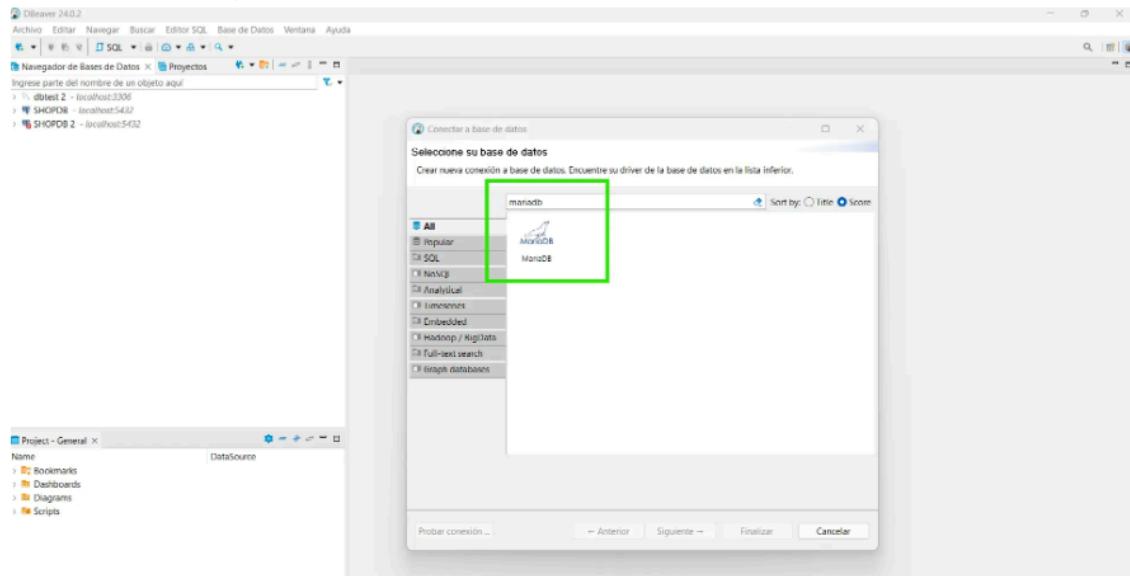
Una vez pulsado el RUN, nos vamos a la sección de contenedores para comprobar que finalmente se haya creado:

Name	Container ID	Image	Port(s)	CPU (%)	Last started	Actions
giffed_panini	21e84ac3c18c	mariadb:its-ubi9	-	0%	20 hours ago	More
penurious_cartwright	d6038f6d92605	mariadb:its-ubi9	-	0%	20 hours ago	More
bbdd	a381d137534e	mariadb:its-ubi9	3306:3306	0.04%	3 minutes ago	More
practicadockercomposew0r	-	-	-	0%	9 months ago	More

3. Acceso a la base de datos.

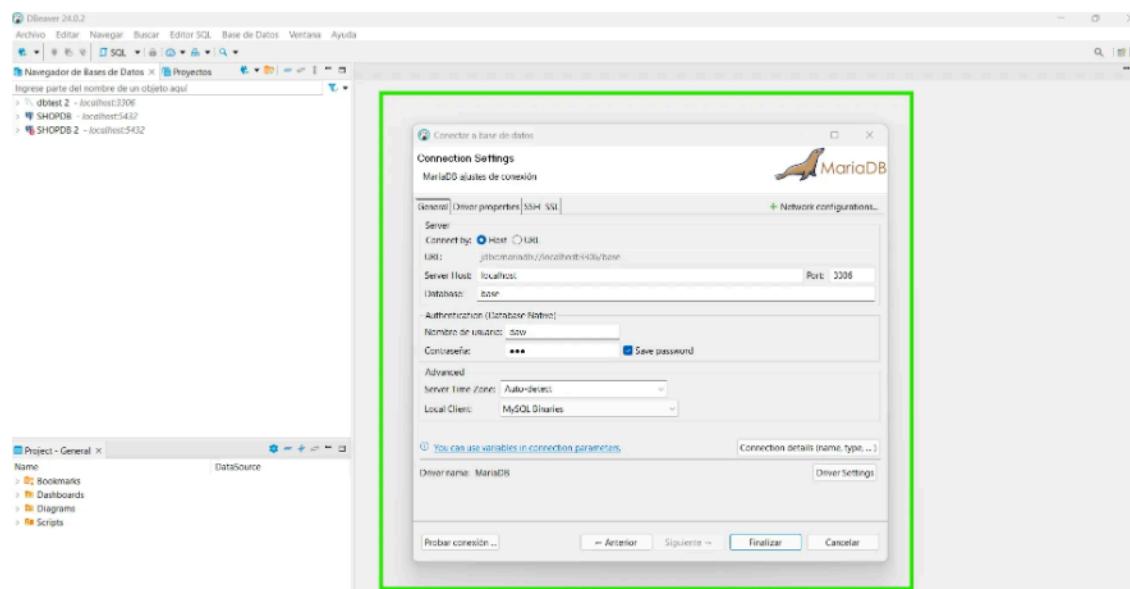
Accede a la base de datos usando una herramienta gráfica, como, por ejemplo dbeaver . Conéctate con el usuario daw . Crea una base de datos y alguna tabla.

- Abrimos **DBeaver** y crear una nueva conexión a MariaDB:

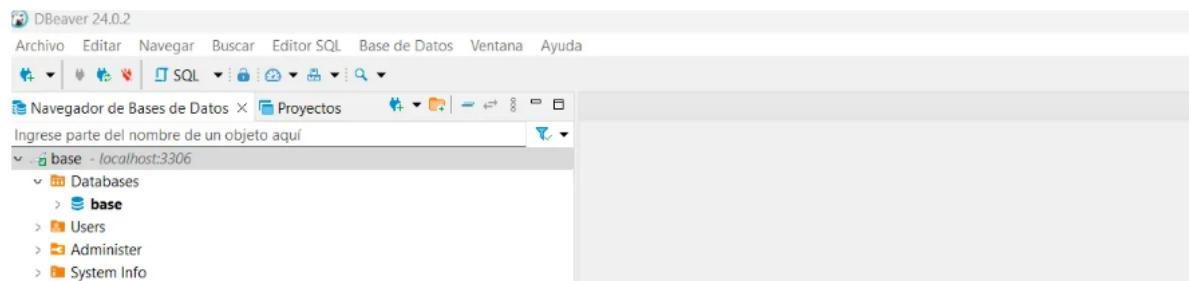


- Configuramos la conexión:

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



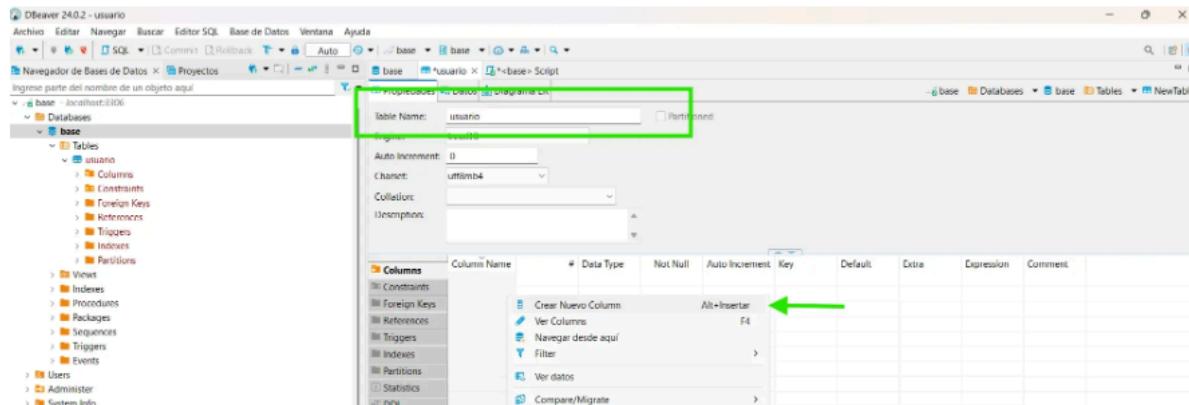
En la columna de la izquierda, podemos comprobar que se ha creado la conexión correctamente:



- Creamos una tabla:

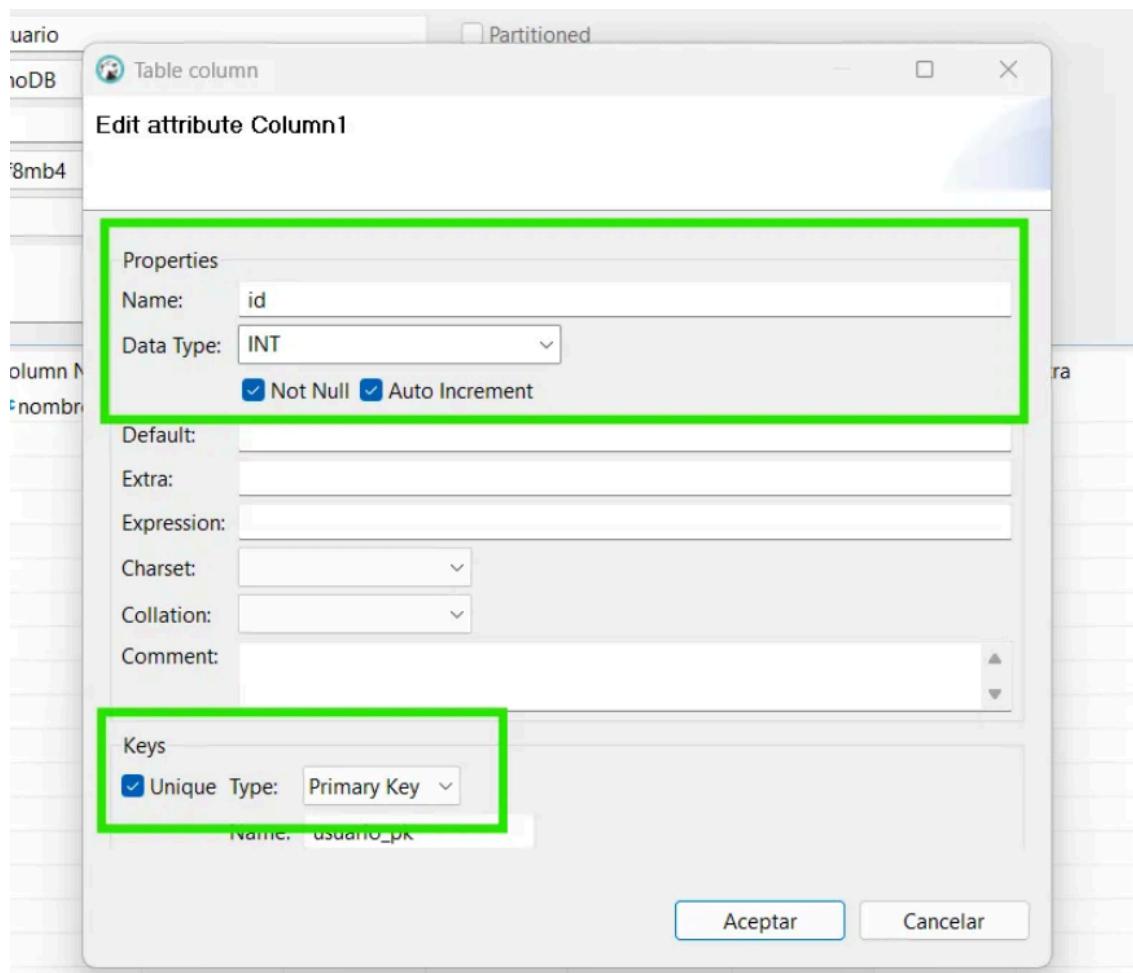
Hacemos click en el botón derecho y pulsamos en "Crear Nueva Columna" para agregar datos:

En *Table Name* introducimos el nombre de la tabla que queramos, en nuestro caso hemos elegido: "usuario".

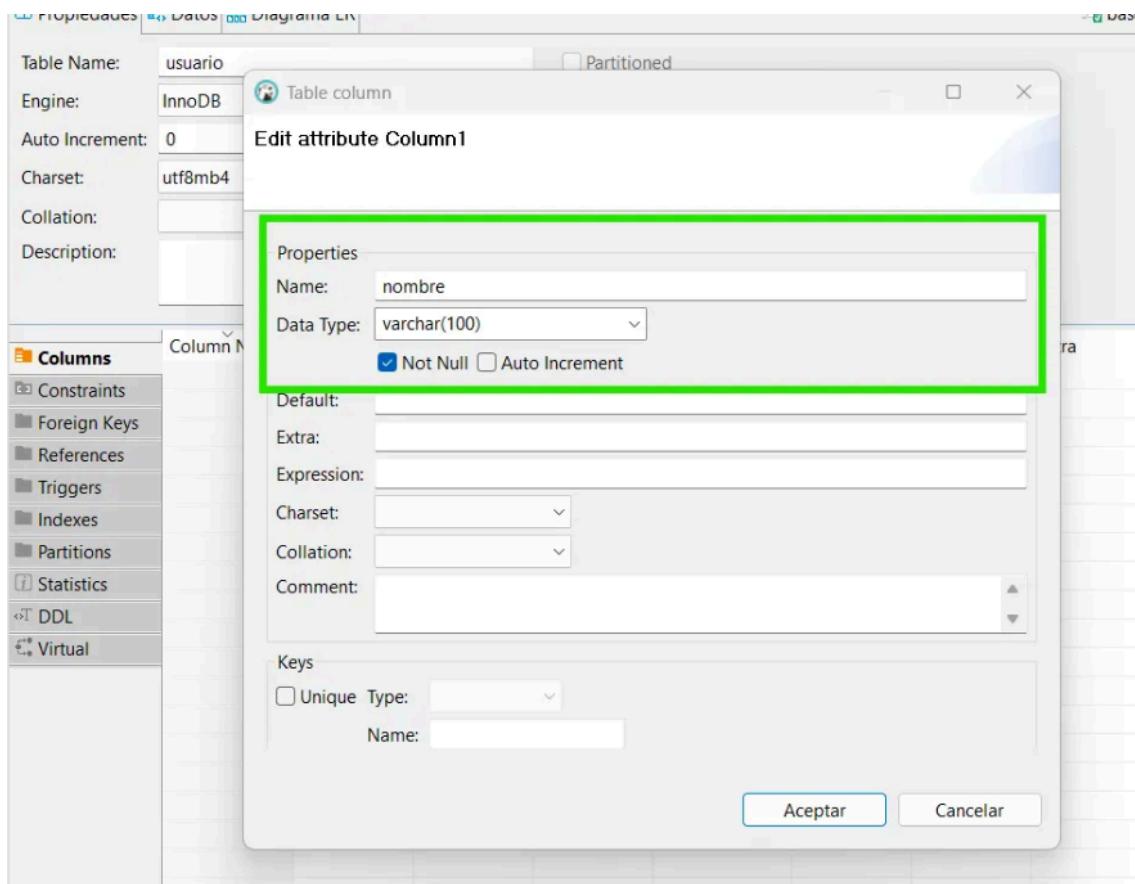


Después hemos creado las columnas para añadir los atributos a nuestra tabla:

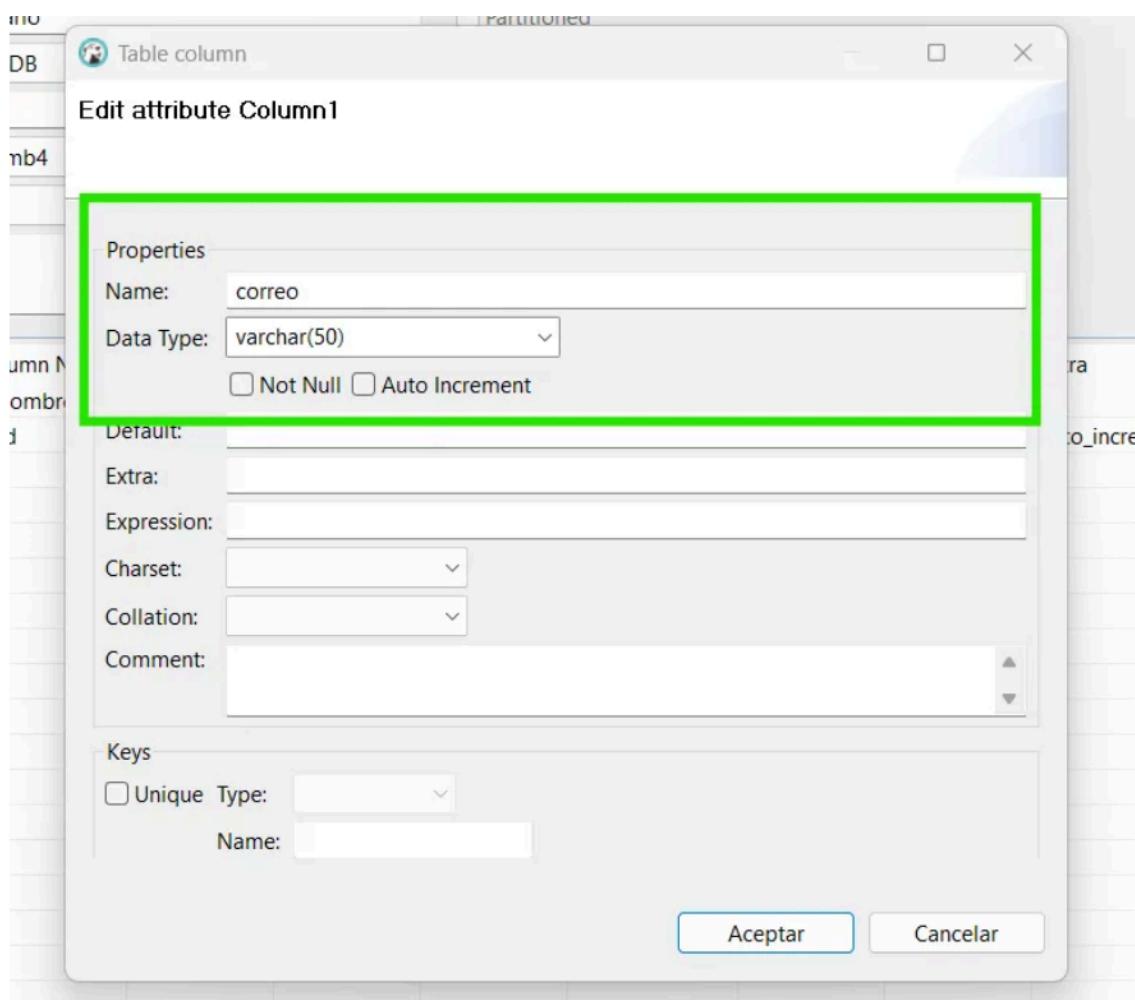
- Un **id**: que es un INT, AUTO_INCREMENT y PK.



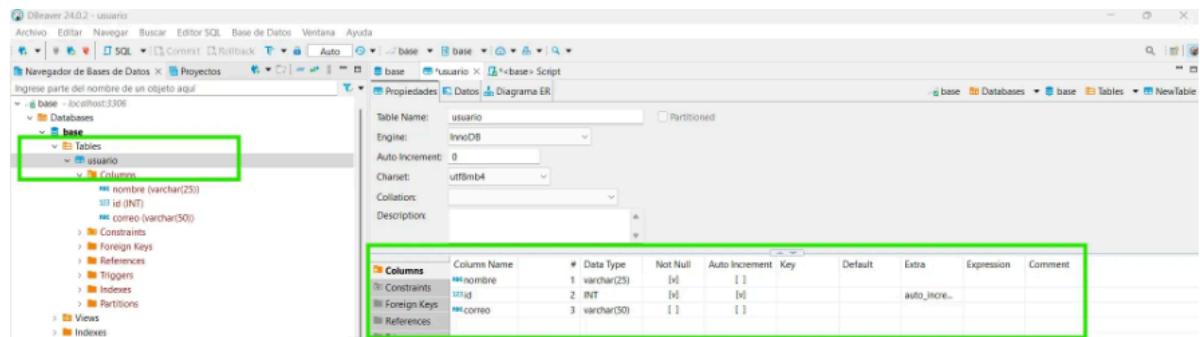
- Un **nombre**: que es un VARCHAR (25) y que no puede ser NULL.



- Un **correo**: que es u VARCHAR de 50.



Comprobamos la creación de la tabla:



4. Borrado del contenedor.

Una vez creada la tabla, intentamos borrar el contenedor:

- Vamos a la pestaña "**Containers**" y buscamos el contenedor bbdd:

Name	Container ID	Image	Port(s)	CPU (%)	Last started	Actions
gifted_panini	21e84ac3c18c	mariadb:10.6		0%	21 hours ago	Stop ⋮ ⋮
nervous_cartwright	d638f6d82605	mariadb:10.6		0%	21 hours ago	Stop ⋮ ⋮
bbdd	a381d137534e	mariadb:10.6	3306:3306	0.02%	34 minutes ago	Stop ⋮ ⋮

- Lo paramos y le damos a eliminar:

Delete container?

The 'bbdd' container is selected for deletion. Any anonymous volumes associated with this container are also deleted.

Cancel
Delete forever

5. Comprobación de la existencia de datos.

Vemos como en Docker Desktop el volumen que contiene los datos no se ha borrado:

- Vamos a la pestaña "**Volumes**" en Docker Desktop. Buscamos datos-mariadb y verificar que sigue ahí:

Volumes Give feedback

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

	Name ↑	Created	Size	Actions
<input type="checkbox"/>	0503e9fad30203f057e35be61d50f957c3faed80ca505a751dc48cc4975eda60	10 months ago	45.9 MB	
<input type="checkbox"/>	3dd453fb4a93e1d22b0c15da8fc06714bed4207c8a605df62f62c08e11bdc6c1	10 months ago	45.5 MB	
<input type="checkbox"/>	4ee9cf14601bbd55bd02ba7003e6523592b995c5e589db04f16a71e8b1d5	10 months ago	46.4 MB	
<input checked="" type="checkbox"/>	a25398a88e58896dcc6d21fa64af3eb3f725a7f3de4ec85ea5584d5888b65d04	21 hours ago	0 Bytes	
<input checked="" type="checkbox"/>	60462009eab72ff8d3a1e003e008d0be41a10e7e707e201dbd433e264d0	93 hours ago	0 Bytes	
<input checked="" type="checkbox"/>	datos-mariadb	50 minutes ago	166.1 MB	
<input type="checkbox"/>	18003c70e24977c108d437c108d32080776367101050727803828839c	10 months ago	43.7 MB	

6. Creación de nuevo contenedor.

Creamos otro contenedor con un servidor de base de datos que use el mismo volumen:

- Llamamos al contenedor **bbdd-2** y utilizamos el mismo volumen que anteriormente:

Run a new container

mariadb:its-ubi9

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	<input type="text" value="datos-mariadb"/> ...	Container path	<input type="text" value="/var/lib/mysql"/> +
-----------	--	----------------	---

Environment variables

Variable	<input type="text" value="MARIADB_ROOT_PASSWORD"/>	Value	<input type="text" value="root"/> -
Variable	<input type="text" value="MARIADB_DATABASE"/>	Value	<input type="text" value="base"/> -
Variable	<input type="text" value="MARIADB_USER"/>	Value	<input type="text" value="daw"/> -
Variable	<input type="text" value="MARIADB_PASSWORD"/>	Value	<input type="text" value="daw"/> +

Cancel **Run**

7. Comprobación de la existencia de la tabla en la base de datos.

Volvemos a **DBeaver** y creamos una nueva conexión a bbdd-2:

Tras darle al botón de 'Finalizar', en la columna de la izquierda, podemos comprobar, que efectivamente, se ha creado una nueva conexión, que mantiene los datos que hemos introducido en el ejercicio anterior, es decir, la tabla de **usuarios** con sus correspondientes **columnas**:

Column Name	Data Type	Not Null	Auto Increment	Key	Default	Extra	Expression	Comment
id	int(11)	Yes	No	Yes				auto_INCREMENT
nombre	varchar(100)	Yes	No	Yes				
correo	varchar(100)	Yes	No	Yes				

8. Eliminación de imagen.

Seguidamente, vamos la pestaña de “Images” en Docker Desktop y buscamos mariadb e intentamos eliminarla:

The screenshot shows the Docker Desktop interface with the "Images" tab selected. A green box highlights the "Images" option in the sidebar. In the main pane, there is a table with columns: Name, Tag, Image ID, Created, Size, and Actions. One row is selected, showing "mariadb" as the name, "lts-ubi9" as the tag, and an image ID starting with "90e4bcac427b". The "Actions" column for this row contains a red "Delete" button and a "Delete" icon with a trash can. A green box highlights the "Delete" button. Below the table, a modal dialog titled "Delete image?" is open, containing the message "The 'mariadb:lts-ubi9' image is selected for deletion." with two buttons: "Cancel" and "Delete forever". A green arrow points from the "Delete" button in the table to the "Delete forever" button in the modal.

¿Qué sucede?

En este caso podemos comprobar que no es posible su eliminación porque hay un contenedor en ejecución.

The screenshot shows the Docker Desktop interface with the "Images" tab selected. A green box highlights the "Images" option in the sidebar. In the main pane, there is a table with columns: Name, Tag, Image ID, Created, Size, and Actions. One row is selected, showing "mariadb" as the name, "lts-ubi9" as the tag, and an image ID starting with "90e4bcac427b". The "Actions" column for this row contains a red "Delete" button and a "Delete" icon with a trash can. A green box highlights the "Delete" button. Below the table, a modal dialog titled "Delete image?" is open, containing the message "Deletion failed" and "Image mariadb:lts-ubi9 is in use. Delete the container that's using it and try again." with a "Close" button. A green box highlights the error message.

9. Eliminación de volumen, imagen y contenedor.

Borramos todo, volumen, imagen y contenedor. Para ello deberemos seguir el siguiente orden, ya que si no, no te dejaría eliminar la imagen de mariadb, puesto que te saldría que está en uso.

- Eliminamos el contenedor `bbdd-2` en **Docker Desktop**:

The screenshot shows the Docker Desktop interface with the "Containers" tab selected. A green box highlights the "Containers" option in the sidebar. In the main pane, there is a table with columns: Name, Container ID, Image, Port(s), CPU (%), Last started, and Actions. One row is selected, showing "bbdd-2" as the name, "6f1e4d9c515f" as the container ID, "mariadb:lts-ubi9" as the image, and port "3306-3306". The "Actions" column for this row contains a red "Delete" button and a "Delete" icon with a trash can. A green box highlights the "Delete" button.

- Vamos a la pestaña "**Volumes**" y borramos datos-mariadb:

The screenshot shows the Docker Desktop interface with the "Volumes" tab selected. A green box highlights the "Volumes" option in the sidebar. In the main pane, there is a table with columns: Name, Image, Last updated, Size, and Actions. One row is selected, showing "datos-mariadb" as the name, "mariadb:lts-ubi9" as the image, and "1 day ago" as the last update. The "Actions" column for this row contains a red "Delete" button and a "Delete" icon with a trash can. A green box highlights the "Delete" button.

- Y por último vamos a "Images" y eliminamos mariadb:

The screenshot shows the Docker desktop application's interface. On the left, there's a sidebar with options like Containers, Images (which is selected), Volumes, Builds, Docker Hub, Docker Scout, and Extensions. The main area is titled 'Images' and shows a list of local images. One image, 'mariadb', is selected. A modal dialog box is overlaid on the screen, containing the text 'Delete image?' and 'The 'mariadb:its-ubi9' image is selected for deletion.' It has two buttons: 'Cancel' (blue) and 'Delete forever' (red). A green arrow points to the 'Delete forever' button. In the top right corner of the main interface, there's a 'Delete' button with a trash icon, which is also highlighted with a green box.

Name	Tag	Image ID	Created	Size	Actions
mariadb	its-ubi9	90e4bcac427b	8 days ago	465.31 MB	Delete

Images		Hub repositories		
Local	Hub repositories			
2.21 GB / 2.9 GB in use	6 images	Last refresh: 23 hours ago	Delete Space to be reclaimed 465.31 MB	
<input type="text" value="Search"/> Filter		Delete image? The 'mariadb:its-ubi9' image is selected for deletion. Cancel Delete forever		
Name		Created	Size	Actions
mariadb	its-ubi9	8 days ago	465.31 MB	Delete
wordpress	latest	9 months ago	685.22 MB	Delete
miapache	latest	10 months ago	255.94 MB	Delete

Ejercicio 3 - contenedores en red: Adminer y MariaDB

La versión que tenemos instalada en nuestros ordenadores no cuenta con *Networks*, por lo que haremos esta tarea desde la terminal a través del uso de comandos.

1. Crea una red bridge `reldb`.

Creamos la red y comprobamos que se creó correctamente.

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

```
andre@Andrea MINGW64 ~/OneDrive/Escritorio/2ºDAW/DAW/DOCKER/Ejercicio Docker/Actividad_Evaluable_Docker/EJERCICIO 3 (ejercicio-3)  
$ docker network create reldb  
b71109d26e88b07828e2acff25183d733798c0ff86b559051094e3791fe674a2
```

```
andre@Andrea MINGW64 ~/OneDrive/Escritorio/2ºDAW/DAW/DOCKER/Ejercicio Docker/Actividad_Evaluable_Docker/EJERCICIO 3 (ejercicio-3)  
$ docker network ls  
NETWORK ID      NAME      DRIVER      SCOPE  
90600e17a714    bridge    bridge      local  
e0bea98cb356    host      host      local  
94f5f1dda64a    none      null      local  
b71109d26e88    reldb    bridge      local
```

2. Crea un contenedor con una imagen de `mariadb` que estará en la red `reldb`. Este contenedor se ejecutará en segundo plano, y será accesible a través del puerto 3306. (Es necesario definir la contraseña del usuario root y un volumen de datos persistente).

Para crear el contenedor con imagen de `mariadb` en nuestra `reldb` tenemos que ejecutar el siguiente comando:

```
$ docker run -d --name mariadb_container --network reldb -p 3307:3306 -e  
MARIADB_ROOT_PASSWORD=root -e MARIADB_DATABASE=base -e MARIADB_USER=daw -e  
MARIADB_PASSWORD=daw -v datos-mariadb:/var/lib/mysql mariadb
```

```
andre@Andrea MINGW64 ~/OneDrive/Escritorio/2ºDAW/DAW/DOCKER/Ejercicio Docker/Actividad_Evaluable_Docker/EJERCICIO 3 (ejercicio-3)  
$ docker run -d --name mariadb_container --network reldb -p 3307:3306 -e MARIADB_ROOT_PASSWORD=root -e MARIADB_DATABASE=base -e MARIADB_USER=daw -e MARIADB_PASSWORD=daw -v datos-mariadb:/var/lib/mysql mariadb  
04a34497c0db68bfd24cc37f264473bcd3adaa86d64d963f496388fe8a5cf2ef
```

- `d`: ejecuta el contenedor en segundo plano.
- `-name mariadb_container`: nombre del contenedor.
- `-network reldb`: conecta a la red `reldb` el contenedor .
- `p 3306:3306`: conecta el puerto **3306** del contenedor *mariadb* al puerto 3307 de nuestro ordenador.
- `e MARIADB_ROOT_PASSWORD=root`: contraseña de *root*.
- `e MARIADB_DATABASE=base`: crea la base de datos *base*.

- `e MARIADB_USER=daw` : crea el usuario `daw` .
- `e MARIADB_PASSWORD=daw` : crea la contraseña `daw` al usuario `daw`.
- `v datos-mariadb:/var/lib/mysql` : usa el volumen `datos-mariadb` para guardar los datos del contenedor.

Una vez creado, comprobamos que está arrancado:

```
docker ps
```

```
andre@Andrea MINGW64 ~/OneDrive/Escritorio/2ºDAW/DAW/DOCKER/Ejercicio Docker/Actividad_Evaluable_Docker/EJERCICIO 3 (ejercicio-3)
$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
04a34497c0db mariadb "docker-entrypoint.s..." 5 seconds ago Up 5 seconds 0.0.0.0:3307->3306/tcp mariadb_container
```

3. Crear un contenedor con *Adminer* o con *phpMyAdmin* que se pueda conectar al contenedor de la BD.

Decidimos usar *Adminer*, ya que es una opción más ligera que *phpMyAdmin*, y creamos el contenedor con ese mismo nombre (*adminer*).

```
andre@Andrea MINGW64 ~/OneDrive/Escritorio/2ºDAW/DAW/DOCKER/Ejercicio Docker/Actividad_Evaluable_Docker/EJERCICIO 3 (ejercicio-3)
$ docker run -d --name adminer --network redbd -p 7777:8080 adminer
d28b8ba8f8710791cb0e1bd232223b539b392acc31af324fa7f74d24064e213f
```

- `docker run`: Inicia un contenedor de Docker.
- `-name adminer`: Establece el nombre del contenedor como `adminer`. Esto facilita su gestión, ya que podrás referirte a él con ese nombre.
- `-network redbd`: conecta el contenedor de Adminer a la red Docker llamada `redbd`, pudiendo así Adminer comunicarse con otros contenedores en esa misma red.
- `p 7777:8080`: expone el puerto 8080 del contenedor (que es el puerto predeterminado de Adminer) al puerto 7777 en nuestra máquina local. Podremos acceder a Adminer en `http://localhost:7777`.
- `d`: ejecuta el contenedor en 2º plano.
- `adminer`: imagen que se usará para crear el contenedor, en este caso, la imagen oficial de Adminer, además, ponemos `adminer` como nombre de contenedor
- Una vez ejecutado, accedemos a Adminer desde nuestro navegador: <http://localhost:7777>



4. Desde la interfaz gráfica, crear una base de datos y una tabla en el servidor de base de datos.

Al acceder a *Adminer* desde nuestro navegador lo primero que debemos hacer es crear una base de datos, a la cual llamaremos `base`.

The screenshot shows the Adminer interface at `localhost:7777/?server=mariadb_container&username=daw`. The main title is "Seleccionar Base de datos". Below it, there's a navigation bar with "Crear Base de datos" (highlighted with a red box), "Privilegios", "Lista de procesos", "Variables", and "Estado". Version information: "Versión MySQL: 11.6.2-MariaDB-ubuntu2404 a través de la extensión de PHP MySQLi". Logged in as "daw@172.18.0.3". On the left, there's a sidebar with "Adminer 4.8.1", "DB: [dropdown]", "Comando SQL", "Importar", and "Exportar". The main content area is titled "Crear Base de datos". It has a form with a "base" input field (highlighted with a green box), a dropdown for character set ("utf8mb4_general_ci"), and buttons for "Guardar" and "+". A green arrow points from the "base" input field towards the "Guardar" button.

Una vez creada nuestra base de datos ya podemos crear una tabla, a la cual llamaremos `sandra_y_andrea`. Su creación es bastante sencilla e intuitiva.

The screenshot shows the Adminer interface at `localhost:7777/?server=mariadb_container&username=daw&db=base`. The main title is "Base de datos: base". Below it, there's a navigation bar with "Modificar Base de datos", "Esquema de base de datos", and "Privilegios". The main content area is titled "Base de datos: base". It has sections for "Tablas y vistas" (with a message "No existen tablas."), "Procedimientos" (with "Crear procedimiento" and "Crear función"), and "Eventos" (with "Crear Evento"). On the left, there's a sidebar with "Adminer 4.8.1", "DB: [dropdown] base", "Comando SQL", "Importar", "Exportar", and "Crear tabla". A green arrow points from the "Crear tabla" button in the sidebar towards the "Crear tabla" button in the "Tablas y vistas" section.

localhost:7777/?server=mariadb_container&username=daw&db=base&create=

Idioma: Español

MySQL » mariadb_container » base » Crear tabla

Adminer 4.8.1

DB: base

Comando SQL Importar Exportar Crear tabla

No existen tablas.

Crear tabla

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

Nombre de columna	Tipo	Longitud	Opciones	NULL	AI?	+		
	int			<input type="checkbox"/>	<input type="radio"/>			

Incremento automático: Valores predeterminados Comentario

Guardar Particionar por

localhost:7777/?server=mariadb_container&username=daw&db=base&create=

Idioma: Español

MySQL » mariadb_container » base » Crear tabla

Adminer 4.8.1

DB: base

Comando SQL Importar Exportar Crear tabla

No existen tablas.

Crear tabla

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

Nombre de columna	Tipo	Longitud	Opciones	NULL	AI?	+		
asignatura	varchar	20	(colación)	<input type="checkbox"/>	<input type="radio"/>			
nota	int	6		<input type="checkbox"/>	<input type="radio"/>			
media	int	6		<input type="checkbox"/>	<input checked="" type="radio"/>			
	int			<input type="checkbox"/>	<input type="radio"/>			

Incremento automático: Valores predeterminados Comentario

Guardar Particionar por



Una vez creada, nuestra base de datos se ve la siguiente manera:

localhost:7777/?server=mariadb_container&username=daw&db=base&table=sandra_y_andrea

Idioma: Español

MySQL » mariadb_container » base » Tabla: sandra_y_andrea

Adminer 4.8.1

DB: base

Comando SQL Importar Exportar Crear tabla

registros sandra_y_andrea

Tabla: sandra_y_andrea

Tabla creada. 10:52:53 Comando SQL

Visualizar contenido Mostrar estructura Modificar tabla Nuevo Registro

Columna	Tipo	Comentario
asignatura	varchar(20)	
nota	int(6)	
media	int(6) Incremento automático	

Índices

PRIMARY media

Modificar índices

Claves externas

Agregar clave externa

Disparadores

Agregar disparador

MySQL » mariadb_container > Base de datos: base

Base de datos: base

Modificar Base de datos Esquema de base de datos Privilegios

Tablas y vistas

Tabla	Motor?	Colación?	Longitud de datos?	Longitud de índice?	Espacio libre?	Incremento automático?	Registros?	Comentario?
sandra_y_andrea	InnoDB	utf8mb4_ucs2_ci	16 384	0	0		1	
1 en total	InnoDB	utf8mb4_ucs2_ci	16 384	0	0		0	

Selected (0)

Analizar Optimizar Comprobar Reparar Vaciar Eliminar

Mover a otra base de datos: base Mover Copiar overwrite

Crear tabla Crear vista

Procedimientos

Crear procedimiento Crear función

Eventos

Crear Evento

Ahora que vemos que todo funciona, ya podemos borrar los contenedores la red y los volúmenes utilizados.

```
$ docker container stop mariadb_container adminer
$ docker container rm mariadb__container adminer
$ docker volume rm datos-mariadb
$ docker network rm redbd
```

Detener los contenedores:

```
andre@Andrea MINGW64 ~/OneDrive/Escritorio/2ºDAW/DAW/DOCKER/Ejercicio Docker/Actividad_Evaluable_Docker/EJERCICIO 3 (ejercicio-3)
$ docker container stop mariadb_container adminer
mariadb_container
adminer
```

Eliminar los contenedores:

```
andre@Andrea MINGW64 ~/OneDrive/Escritorio/2ºDAW/DAW/DOCKER/Ejercicio Docker/Actividad_Evaluable_Docker/EJERCICIO 3 (ejercicio-3)
$ docker container rm mariadb_container adminer
mariadb_container
adminer
```

Eliminar el volumen:

```
andre@Andrea MINGW64 ~/OneDrive/Escritorio/2ºDAW/DAW/DOCKER/Ejercicio Docker/Actividad_Evaluable_Docker/EJERCICIO 3 (ejercicio-3)
$ docker volume rm datos-mariadb
datos-mariadb
```

Eliminar la red:

```
andre@Andrea MINGW64 ~/OneDrive/Escritorio/2ºDAW/DAW/DOCKER/Ejercicio Docker/Actividad_Evaluable_Docker/EJERCICIO 3 (ejercicio-3)
$ docker network rm redbd
redbd
```

Ejercicio 4 - Docker compose

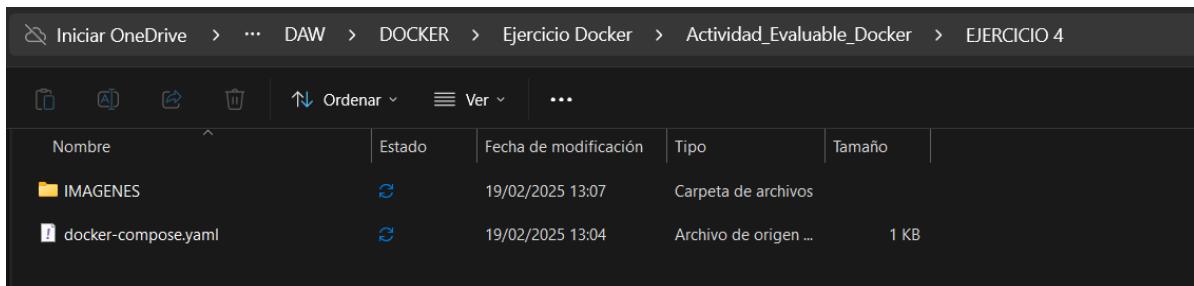
En esta actividad se nos pide desplegar la aplicación htop utilizando docker-compose.

`htop` es una herramienta de monitoreo interactivo de procesos para sistemas Linux/Unix. Se utiliza como interfaz visual y destaca por su fácil uso para ver los procesos en ejecución, el uso de la CPU, la memoria y otras estadísticas del sistema. A diferencia de otros comandos como `top`, `htop` permite una navegación más sencilla a través de los procesos y permite realizar acciones como matar procesos directamente desde su interfaz.

En esta tarea, vamos a ejecutar `**htop**` dentro de un contenedor Docker a través de Docker Compose para facilitar el despliegue.

En primer lugar, creamos el archivo a través de comandos y comprobamos que se creó sin problema:

```
andre@Andrea MINGW64 ~/OneDrive/Escritorio/2ºDAW/DAW/DOCKER/Ejercicio Docker/Actividad_Evaluable_Docker/EJERCICIO 4 (ejercicio-4)
$ touch docker-compose.yaml
```

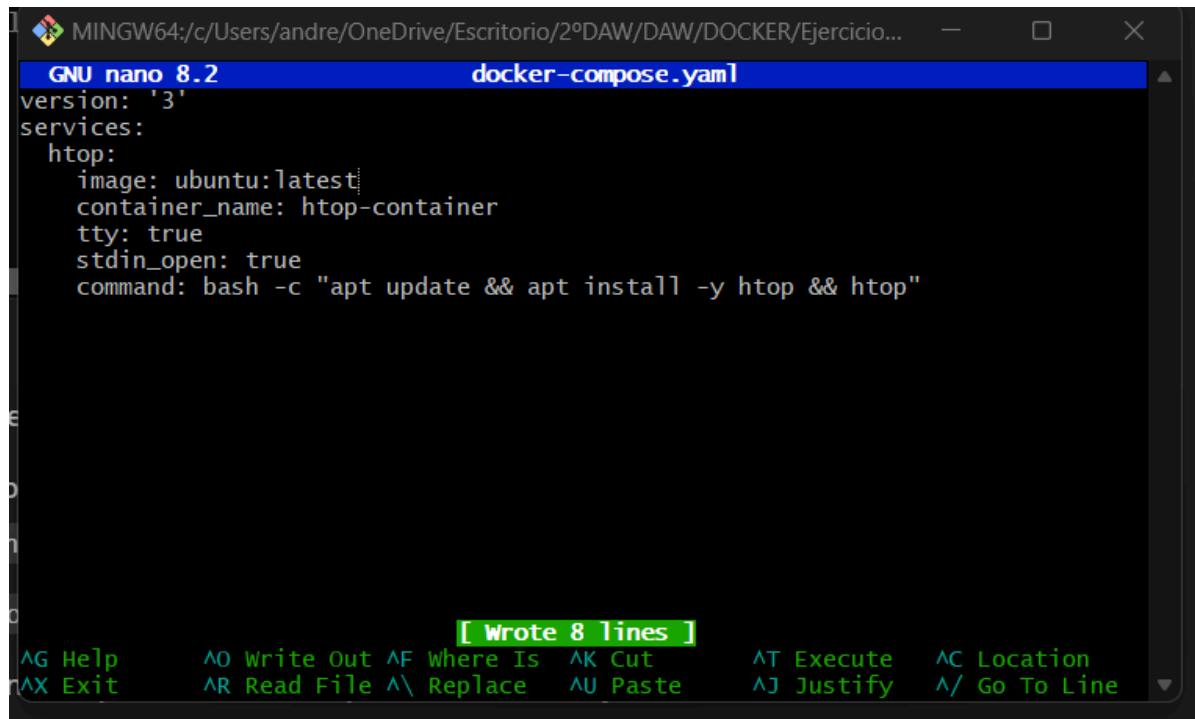


Una vez creado, lo editamos con el editor `nano`.

```
$ nano docker-compose.yaml
```

```
andre@Andrea MINGW64 ~/OneDrive/Escritorio/2ºDAW/DAW/DOCKER/Ejercicio Docker/Actividad_Evaluable_Docker/EJERCICIO 4 (ejercicio-4)
$ nano docker-compose.yaml
```

```
version: '3'
services:
  htop:
    image: ubuntu:latest
    container_name: htop-container
    tty: true
    stdin_open: true
    command: bash -c "apt update && apt install -y htop && htop"
```



```
GNU nano 8.2 docker-compose.yaml
version: '3'
services:
  htop:
    image: ubuntu:latest
    container_name: htop-container
    tty: true
    stdin_open: true
    command: bash -c "apt update && apt install -y htop && htop"
```

[Wrote 8 Lines]

AG Help ^O Write Out ^F Where Is ^K Cut ^T Execute ^C Location
AX Exit ^R Read File ^\ Replace ^U Paste ^J Justify ^/ Go To Line

Para levantar el contenedor usamos el comando `docker-compose up -d`, abriendo el terminal en dicho archivo, siendo el `docker-compose` para gestionar varios contenedores y `up` para iniciarlos siendo `-d` para ejecutarlos en segundo plano.

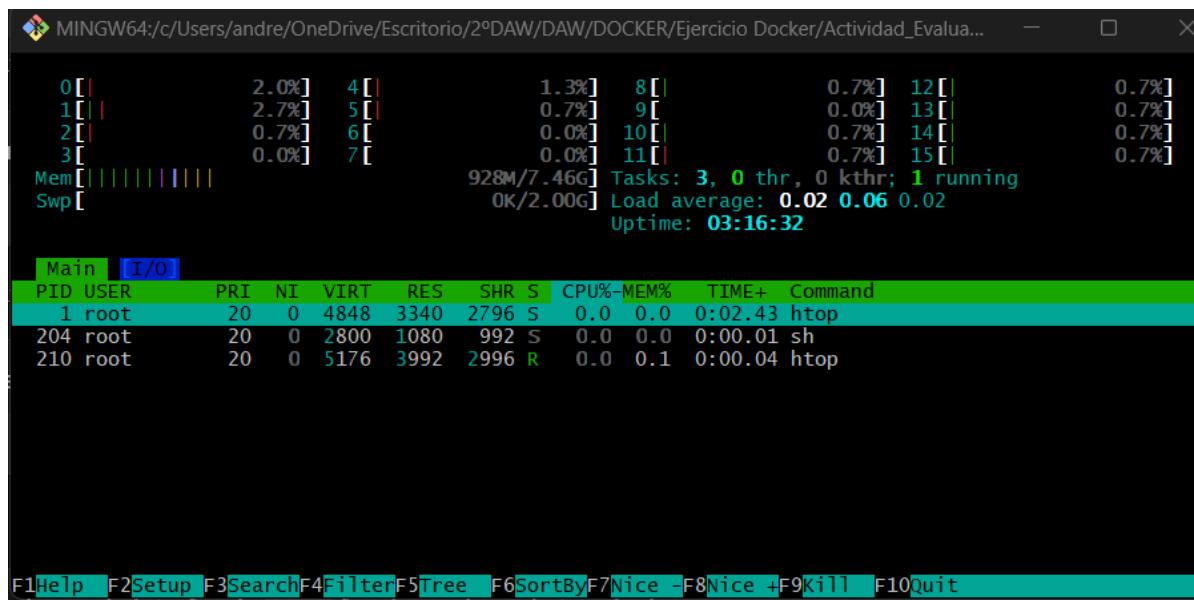
```
$ docker-compose up -d
```

```
andre@Andrea MINGW64 ~/OneDrive/Escritorio/2ºDAW/DAW/DOCKER/Ejercicio Docker/Actividad_Evaluable_Docker/EJERCICIO 4 (ejercicio-4)
$ docker-compose up -d
time="2025-02-19T13:11:44+01:00" level=warning msg="C:\\\\Users\\\\andre\\\\OneDrive\\\\Escritorio\\\\2ºDAW\\\\DAW\\\\DOCKER\\\\Ejercicio Docker\\\\Actividad_Evaluable_Docker\\\\EJERCICIO 4\\\\docker-compose.yaml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion"
[+] Running 2/2
  ✓ htop Pulled
    ✓ 5a7813e071bf Already exists
[+] Running 2/2
  ✓ Network ejercicio4_default Created
  ✓ Container htop-container Started
```

Ahora es hora de poner en marcha el contenedor y verlo en acción. Para ello lo primero que hacemos es acceder a él a través del siguiente comando:

```
$ docker exec -it htop-container sh
# htop
```

Esto nos permite ver una interfaz de `htop` funcionando dentro de nuestro contenedor.



Una vez terminado el ejercicio ya podemos detener y eliminar el contenedor si así lo queremos:

```
$ docker-compose down
```

Ejercicio 5 - imagen con Dockerfile

Ejercicio 5 - imagen con Dockerfile

PARTE I. ARRANQUE Y AUTOMATIZACIÓN DE IMAGEN EN DOCKERFILE

PARTE II. DESCARGA DE IMAGEN Y CREACIÓN DEL CONTENEDOR

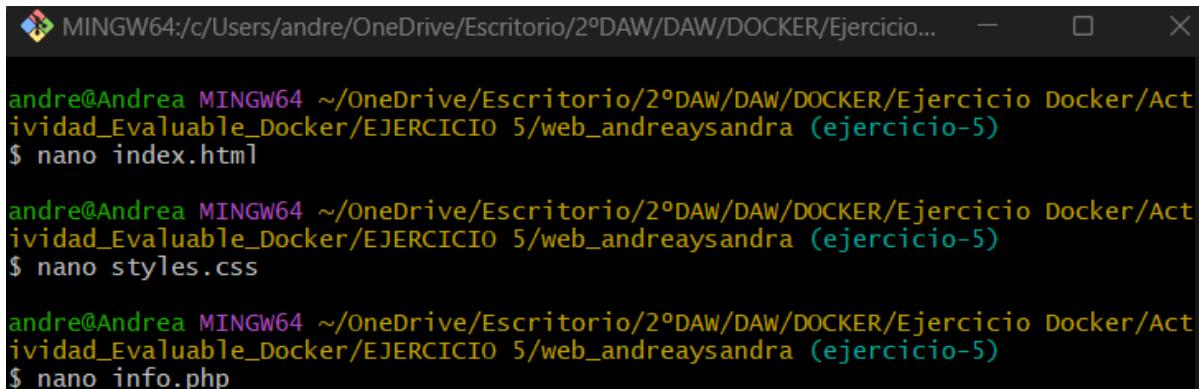
PARTE I. ARRANQUE Y AUTOMATIZACIÓN DE IMAGEN EN DOCKERFILE

En este apartado se nos pide arrancar un contenedor que ejecute una instancia de la imagen php:7.4-apache , que se llame web y que sea accesible desde un navegador en el puerto 8000, en nuestro caso, 8800.

```
docker run -d --name web -p 8800:80 php:7.4-apache
```

Creamos el sitio web y, una vez creado, hacemos lo mismo con el index, hoja de estilos y el script PHP.

```
mkdir web_andreaysandra  
cd web_andreaysandra
```



```
andre@Andrea MINGW64 ~/OneDrive/Escritorio/2ºDAW/DAW/DOCKER/Ejercicio... — X  
andre@Andrea MINGW64 ~/OneDrive/Escritorio/2ºDAW/DAW/DOCKER/Ejercicio Docker/Actividad_Evaluable_Docker/EJERCICIO 5/web_andreaysandra (ejercicio-5)  
$ nano index.html  
  
andre@Andrea MINGW64 ~/OneDrive/Escritorio/2ºDAW/DAW/DOCKER/Ejercicio Docker/Actividad_Evaluable_Docker/EJERCICIO 5/web_andreaysandra (ejercicio-5)  
$ nano styles.css  
  
andre@Andrea MINGW64 ~/OneDrive/Escritorio/2ºDAW/DAW/DOCKER/Ejercicio Docker/Actividad_Evaluable_Docker/EJERCICIO 5/web_andreaysandra (ejercicio-5)  
$ nano info.php
```

Creamos index.html y lo editamos:

```
nano index.html  
<!DOCTYPE html>  
<html lang="es">  
<head>  
    <meta charset="UTF-8">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>Probando PHP</title>  
    <link rel="stylesheet" href="styles.css">  
</head>  
<body>  
    <h1>Bienvenidos a nuestra WEB</h1>  
    <form action="script.php" method="post">  
        <h2>Hola, somos Sandra y Andrea</h2>  
        <p>Somos estudiantes de 2º de DAW y estamos probando nuestra página usa>  
</body>  
</html>
```

The screenshot shows a terminal window titled "GNU nano 8.2" with the file "index.html" open. The content of the file is:

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Probando PHP</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <h1>Bienvenidos a nuestra WEB</h1>
    <form action="script.php" method="post">
        <h2>Hola, somos Sandra y Andrea</h2>
        <p>Somos estudiantes de 2º de DAW y estamos probando nuestra página usando PHP.</p>
    </form>
</body>
</html>
```

At the bottom of the terminal, there is a menu bar with the following options:

- File Name to Write: index.html
- ^G Help
- ^C Cancel
- M-D DOS Format
- M-M Mac Format
- M-A Append
- M-P Prepend
- M-B Backup File
- ^T Browse

Ahora creamos el css:

```
nano styles.css
body {
    font-family: Arial, sans-serif;
    background-color: #7d2181;
    text-align: center;
    margin: 0;
    padding: 0;
}

h1 {
    color: black;
    margin-top: 20px;
}

p {
    font-size: 18px;
    color: #555;
    background: black;
    padding: 10px;
    border-radius: 8px;
    display: inline-block;
```

GNU nano 8.2 styles.css Modified

```
body {  
    font-family: Arial, sans-serif;  
    background-color: #7d2181;  
    text-align: center;  
    margin: 0;  
    padding: 0;  
}  
  
h1 {  
    color: black;  
    margin-top: 20px;  
}  
  
p {  
    font-size: 18px;  
    color: #555;  
    background: black;  
    padding: 10px;  
    border-radius: 8px;  
    display: inline-block;  
}
```

File Name to Write: styles.css

AG Help M-D DOS Format M-A Append M-B Backup File
AC Cancel M-M Mac Format M-P Prepend AT Browse

Y, finalmente, el script PHP:

```
nano 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>";  
?>
```

GNU nano 8.2 info.php Modified

```
<?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>";  
?>
```

File Name to Write: info.php

AG Help M-D DOS Format M-A Append M-B Backup File
AC Cancel M-M Mac Format M-P Prepend AT Browse

Una vez creados, copiamos los archivos en el contenedor, paso que hay que repetir cada vez que se modifique uno de ellos:

```
docker cp index.html web:/var/www/html/
docker cp styles.css web:/var/www/html/
docker cp info.php web:/var/www/html/
```

```
andre@Andrea MINGW64 ~/OneDrive/Escritorio/2ºDAW/DAW/DOCKER/Ejercicio Docker/Actividad_Evaluable_Docker/EJERCICIO 5/web_andreaysandra (ejercicio-5)
$ docker cp index.html web:/var/www/html/
Successfully copied 2.05kB to web:/var/www/html/

andre@Andrea MINGW64 ~/OneDrive/Escritorio/2ºDAW/DAW/DOCKER/Ejercicio Docker/Actividad_Evaluable_Docker/EJERCICIO 5/web_andreaysandra (ejercicio-5)
$ docker cp styles.css web:/var/www/html/
Successfully copied 2.05kB to web:/var/www/html/

andre@Andrea MINGW64 ~/OneDrive/Escritorio/2ºDAW/DAW/DOCKER/Ejercicio Docker/Actividad_Evaluable_Docker/EJERCICIO 5/web_andreaysandra (ejercicio-5)
$ docker cp info.php web:/var/www/html/
Successfully copied 2.05kB to web:/var/www/html/
```

Una vez realizados estos pasos, es hora de comprobar en el navegador que se están aplicando correctamente. Visitamos <http://localhost:8800> y <http://localhost:8800/info.php>.



Información

Hoy es 19/02/2025

El mes es: February

Hora: 23:37:55

Ahora vamos a proceder con la automatización en Docker. Para subir la imagen a la cuenta de Docker hay que crear un fichero Dockerfile y editararlo.

```
$ touch Dockerfile
$ nano Dockerfile
```

```
andre@Andrea MINGW64 ~/OneDrive/Escritorio/2ºDAW/DAW/DOCKER/Ejercicio Docker/Actividad_Evaluable_Docker/EJERCICIO 5/web_andreaysandra (ejercicio-5)
$ touch Dockerfile

andre@Andrea MINGW64 ~/OneDrive/Escritorio/2ºDAW/DAW/DOCKER/Ejercicio Docker/Actividad_Evaluable_Docker/EJERCICIO 5/web_andreaysandra (ejercicio-5)
$ nano Dockerfile
```

```

# utiliza la imagen oficial de PHP con Apache como base
FROM php:7.4-apache

# Copia los archivos al contenedor
COPY index.html /var/www/html/
COPY styles.css /var/www/html/
COPY info.php /var/www/html/

# Expone el puerto 8000 para acceder al contenedor
EXPOSE 8000

# Configura Apache para escuchar en el puerto 8000
CMD ["apache2-foreground"]

```

File Name to Write: Dockerfile

AG Help M-D DOS Format M-A Append M-B Backup File
 AC Cancel M-M Mac Format M-P Prepend AT Browse

Una vez creado y editado, ya podemos crear la imagen.

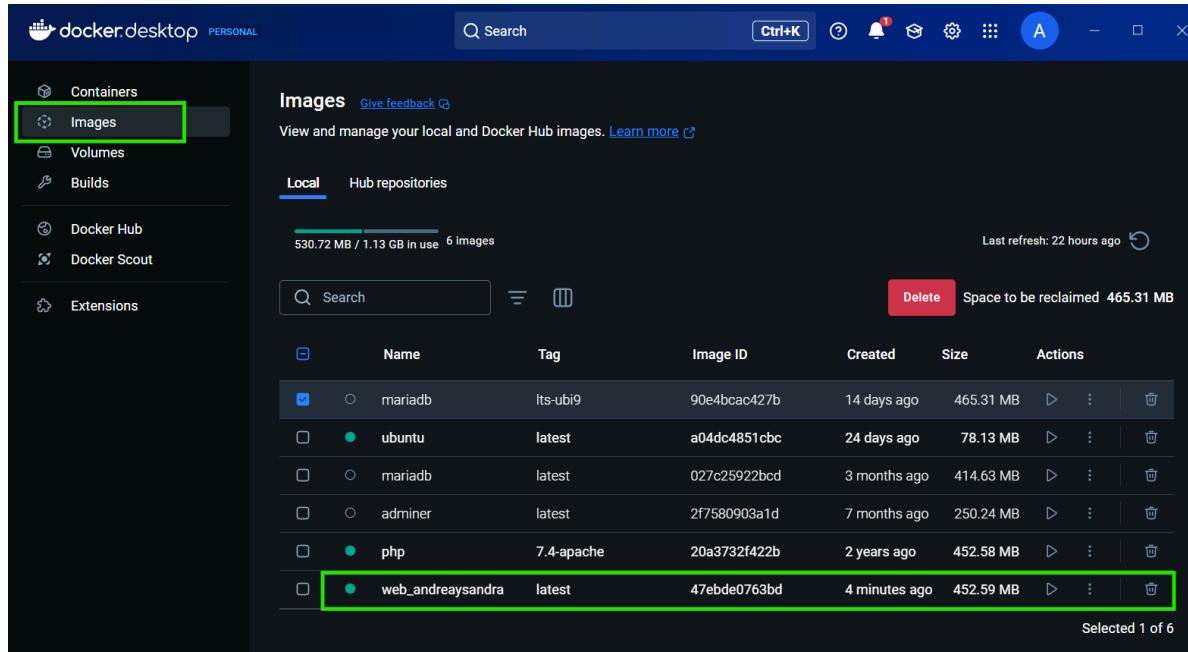
```
$ docker build -t web_andreaysandra .
```

```

andre@Andrea MINGW64 ~/OneDrive/Escritorio/2ºDAW/DAW/DOCKER/Ejercicio Docker/Actividad_Evaluable_Docker/EJERCICIO 5/web_andreaysandra (ejercicio-5)
$ docker build -t web_andreaysandra .
[+] Building 0.7s (9/9) FINISHED                                            docker:desktop-linux
=> [internal] load build definition from Dockerfile                      0.0s
=> => transferring dockerfile: 385B                                         0.0s
=> [internal] load metadata for docker.io/library/php:7.4-apache          0.0s
=> [internal] load .dockerignore                                           0.0s
=> => transferring context: 2B                                         0.0s
=> [internal] load build context                                         0.1s
=> => transferring context: 401B                                         0.0s
=> CACHED [1/4] FROM docker.io/library/php:7.4-apache                      0.0s
=> [2/4] COPY index.html /var/www/html/                                       0.1s
=> [3/4] COPY styles.css /var/www/html/                                      0.1s
=> [4/4] COPY info.php /var/www/html/                                       0.1s
=> exporting to image                                                 0.2s
=> => exporting layers                                              0.1s
=> => writing image sha256:47ebde0763bdeac02aea9b4ddb2ab92473150c088d9af 0.0s
=> => naming to docker.io/library/web_andreaysandra                         0.0s

```

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/02zacjfmj3uy5v9rtm9fit7n5



PARTE II. DESCARGA DE IMAGEN Y CREACIÓN DEL CONTENEDOR

El siguiente paso es la descarga de la imagen y la creación del contenedor con dicha imagen. Para ello, hemos seguido los siguientes pasos:

Actualizamos el repositorio con los cambios realizados para descargarnos la imagen creada por nuestro compañero/a:

```
$git pull origin main
```

```
PS C:\Users\34615\Desktop\2°DAW\DAW\SEGUNDA EVALUACION\UT8. DOCKER\TAREA EVALUABLE DOCKER\EJERCICIO 5\web_andreaysandra> git pull origin main
```

Para asegurarnos de que la imagen se construyó correctamente, listamos las imágenes disponibles ejecutando el siguiente comando:

```
$docker images
```

```
PS C:\Users\34615\Desktop\2°DAW\DAW\SEGUNDA EVALUACION\UT8. DOCKER\TAREA EVALUABLE DOCKER\EJERCICIO 5\web_andreaysandra> docker images
REPOSITORY          TAG      IMAGE ID            CREATED             SIZE
web_andreaysandra  latest   a8442eed1771    58 seconds ago   453MB
mariadb             latest   e211cffc32e1    6 days ago        390MB
wordpress           latest   5fb7d355caa4    9 months ago       685MB
miapache            latest   3f4db31842cd    10 months ago      256MB
postgres            16.2    d60dc4hd84c9    12 months ago      431MB
mysql               5.7     5107333e08a8    14 months ago      501MB
phpmyadmin/phpmyadmin latest   933569f3a9f6    19 months ago      562MB
```

Una vez tenemos la imagen construida, ejecutamos un nuevo contenedor con el siguiente comando:

```
$docker run -d --name web_andrea -p 8800:80 web_andreaysandra
```

```
PS C:\Users\34615\Desktop\2°DAW\DAW\SEGUNDA EVALUACION\UT8. DOCKER\TAREA EVALUABLE DOCKER\EJERCICIO 5\web_andreaysandra> docker run -d --name web_andrea -p 8800:80 web_andreaysandra
6e5642dddea369b410d82b1fd0273db7c624f2d1539d3272a5f2a80a52567d0
```

Para asegurarnos de que el contenedor está en funcionamiento, lo verificamos con el siguiente comando:

```
$docker ps
```

```
ps C:\Users\srujalarro\OneDrive\Documentos\SEGUNDA EVALUACION\UT8. DOCKER\TAREA EVALUABLE DOCKER\EJERCICIO 5\web_andreaysandra> docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS
6e5642ddee3        web_andreaysandra   "docker-php-entrypoi..."   50 seconds ago    Up 48 seconds      8000/tcp, 0.0.0.0:8800->80/tcp   web_andrea
```

Podemos abrir nuestra aplicación de Docker Desktop para comprobar que hemos creado correctamente el contenedor:

Name	Container ID	Image	Port(s)	CPU (%)	Last started
web_andrea	6e5642ddee3	web_andreaysandra	8800:80	0.01%	10 minutes ago

Para comprobar que los pasos han sido correctos, abrimos un navegador y pegamos la siguiente url:

- <http://localhost:8800/>

Resultado del acceso al navegador:

Bienvenidos a nuestra WEB

Hola, somos Sandra y Andrea

Somos estudiantes de 2º de DAW y estamos probando nuestra página usando PHP.