

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.

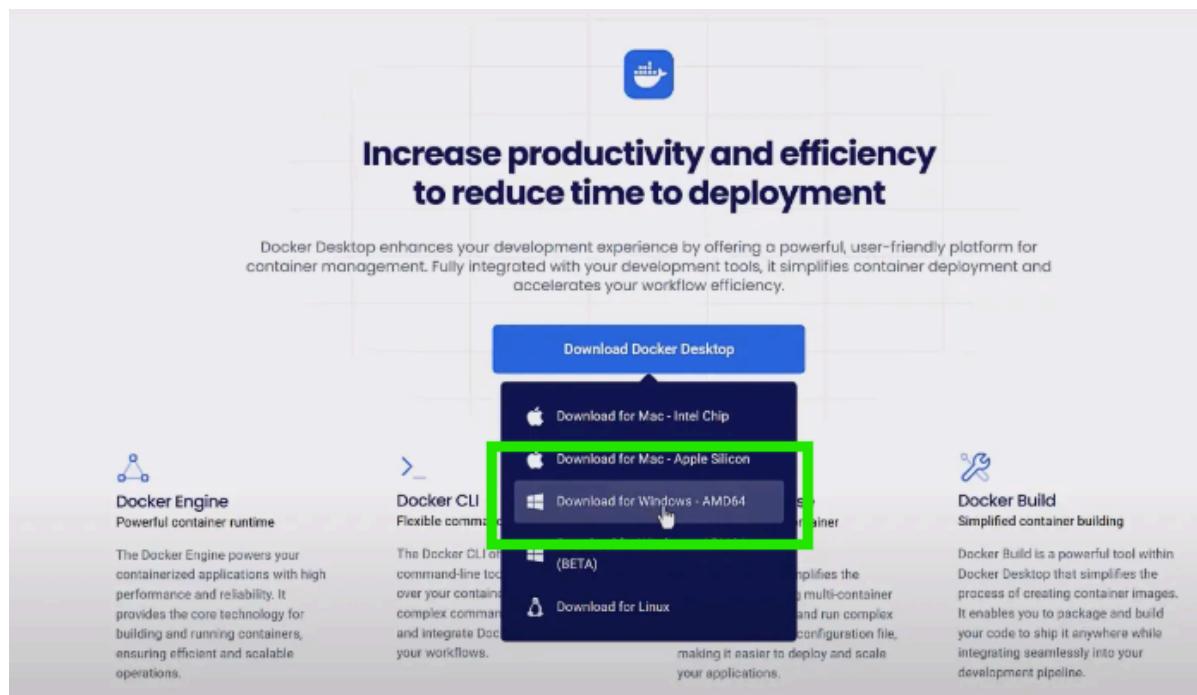
1. Instalación de Docker Desktop

1.1 Requisitos del Sistema

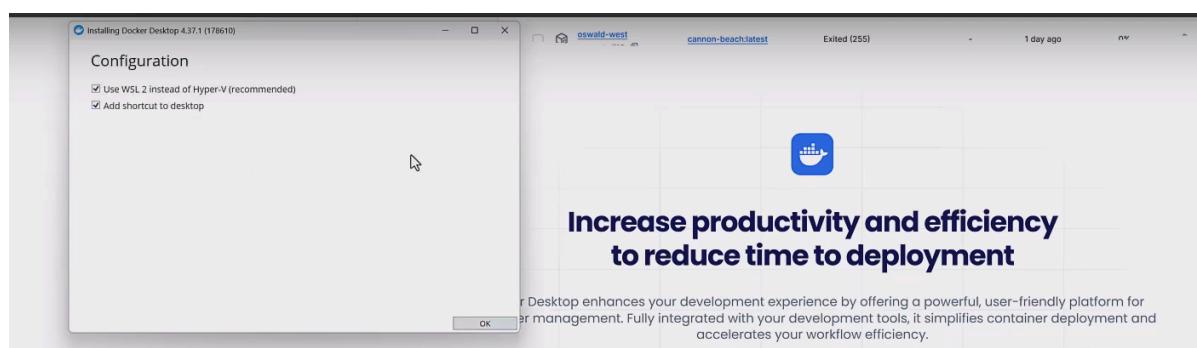
- Windows 10/11 (con WSL2 habilitado) o macOS 10.14+
- Procesador de 64 bits con soporte para virtualización
- Al menos 4 GB de RAM

1.2 Pasos de Instalación

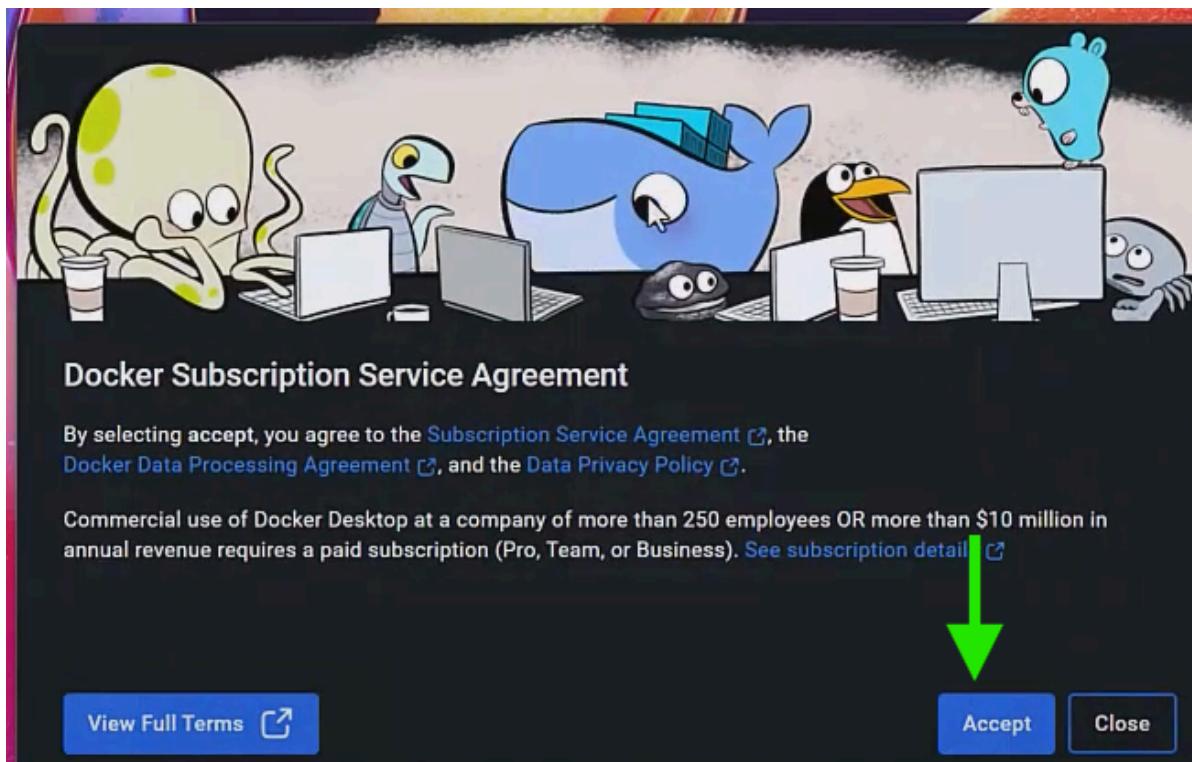
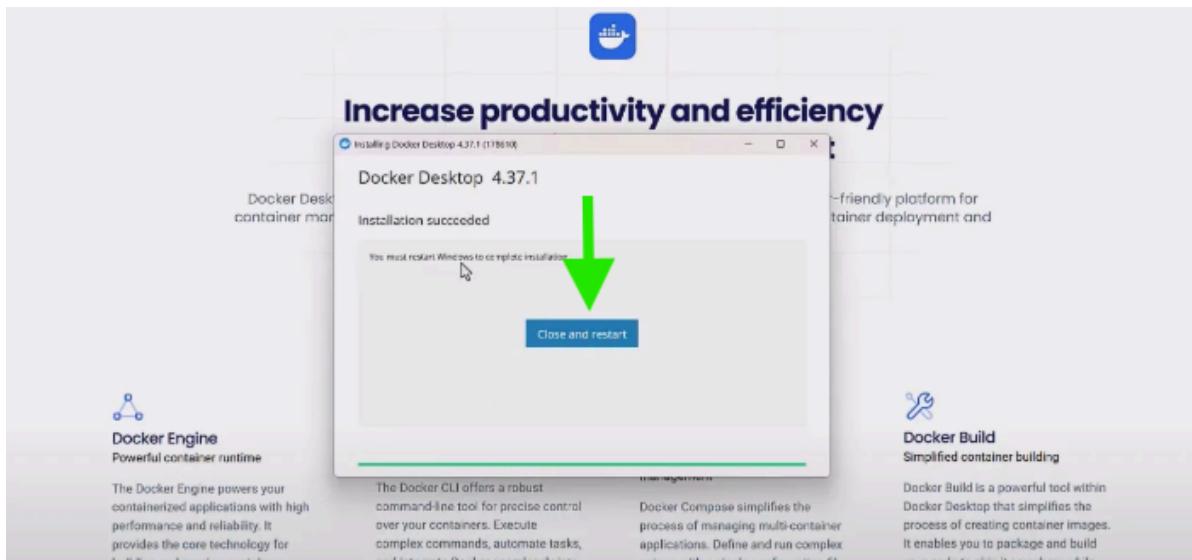
Descargamos Docker Desktop desde el sitio oficial: <https://www.docker.com/products/docker-desktop>.



Ejecutamos el instalador y seguimos las instrucciones en pantalla.



Reiniciamos el sistema si es necesario.



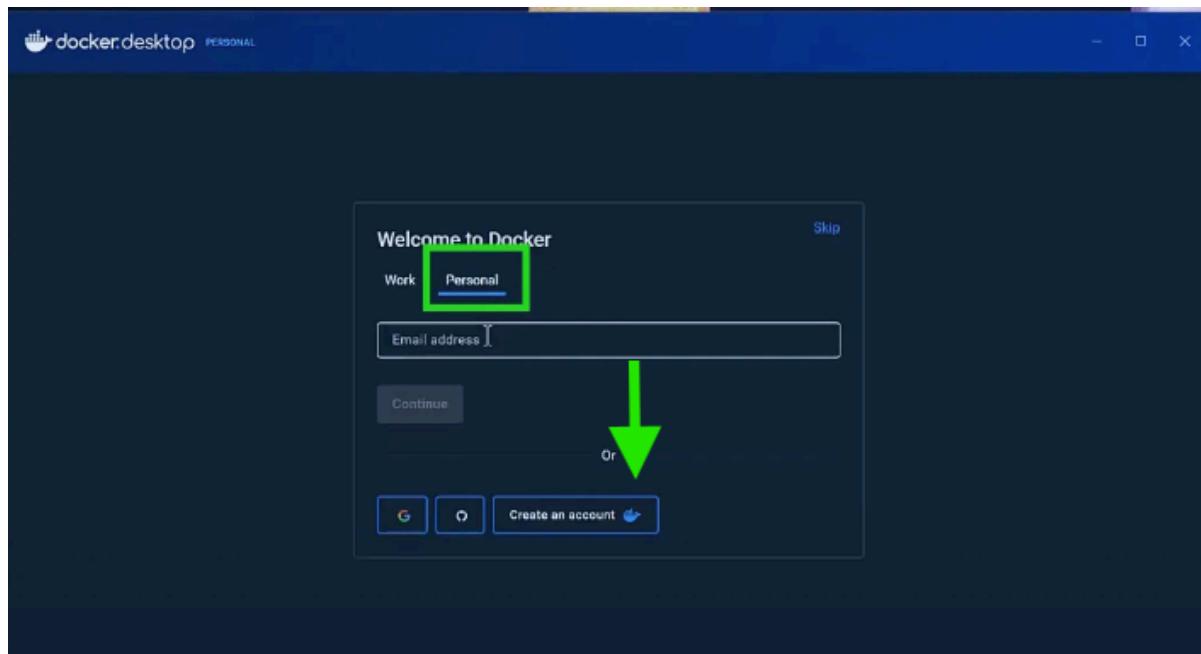
Iniciamos Docker Desktop y verificamos que esté corriendo desde la barra de tareas o con el comando:

```
docker --version
```

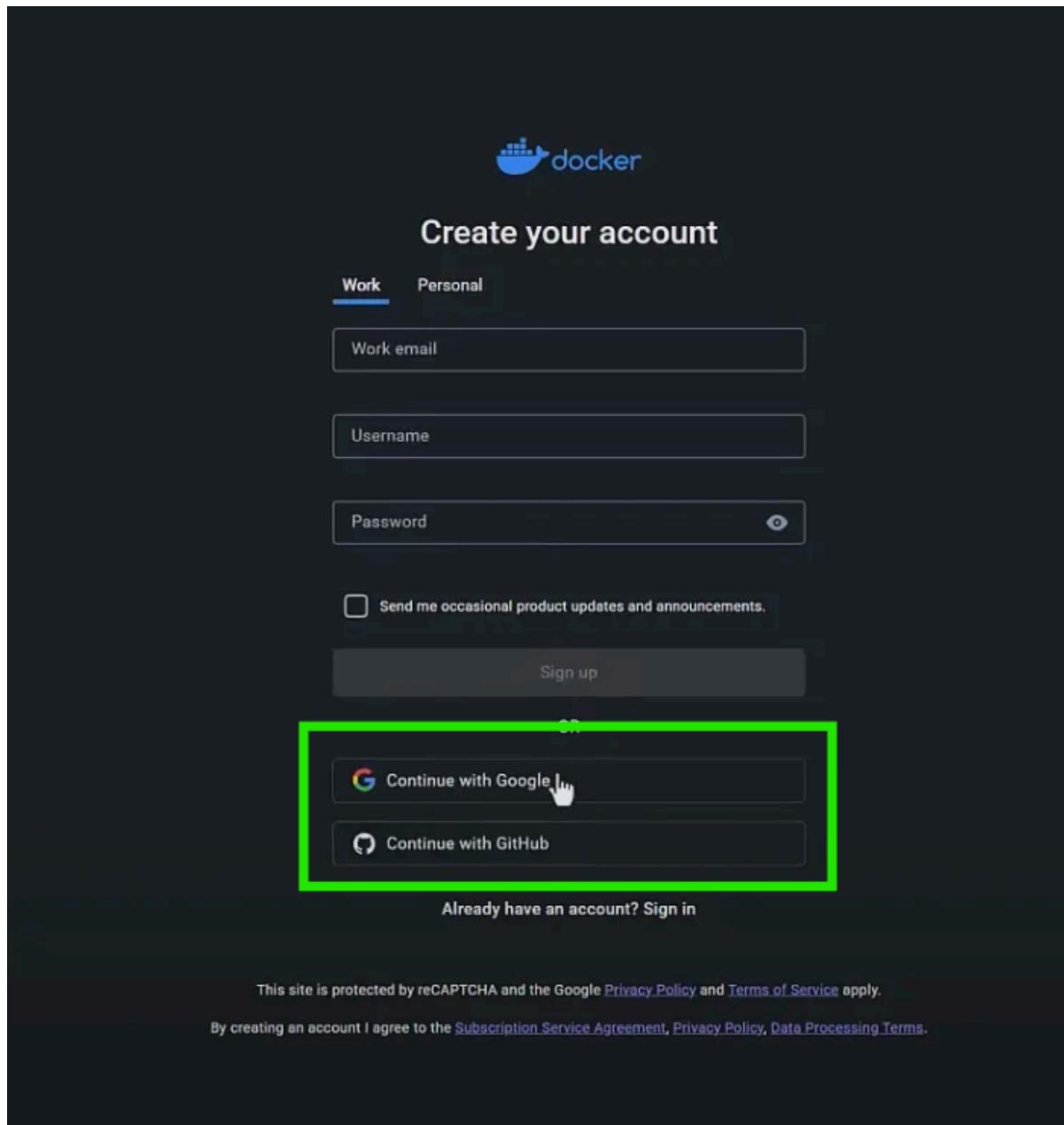
```
C:\Users\34615>docker --version
Docker version 25.0.3, build 4debf41
```

2. Configuración Inicial

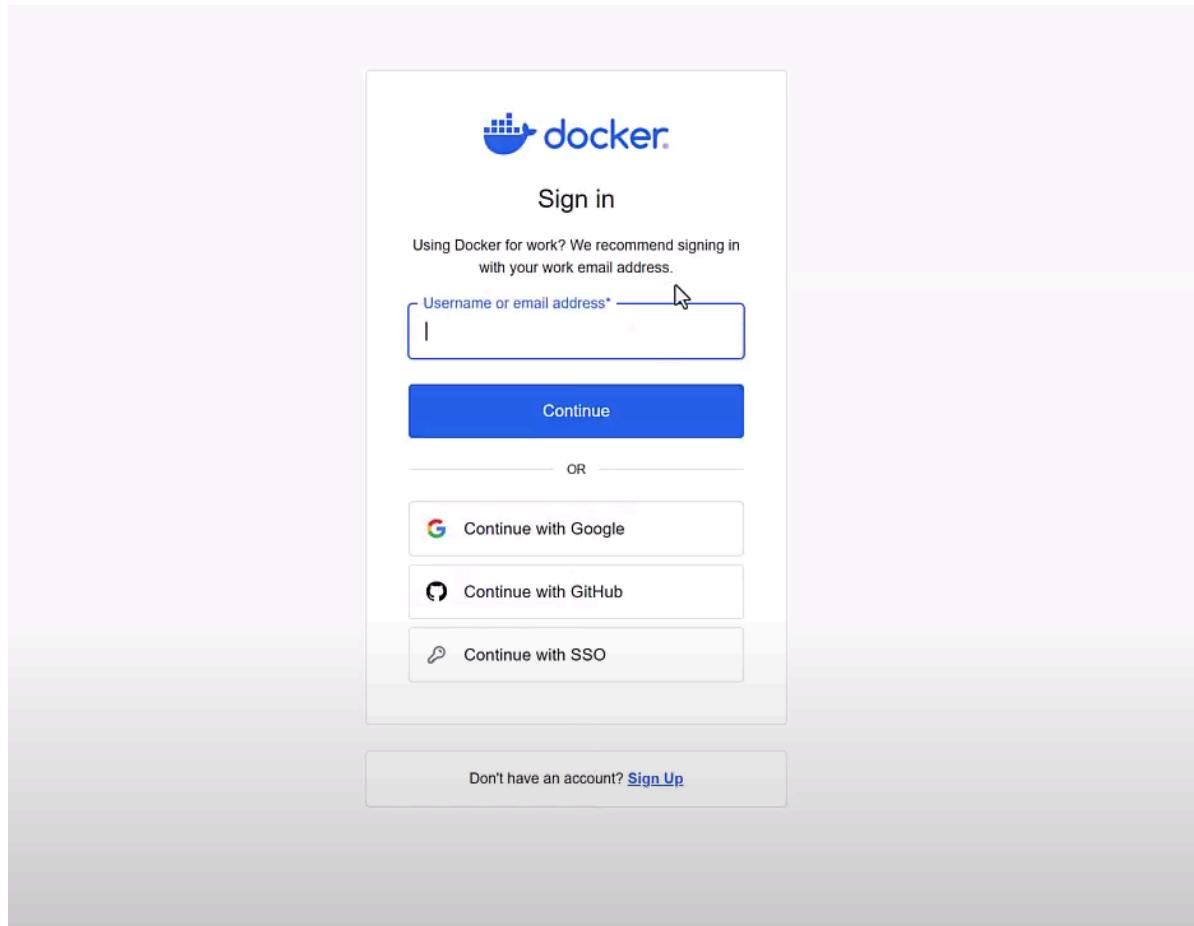
Configuramos Docker, le damos uso de manera “Personal”. Si ya tenemos una cuenta de Docker, debemos introducirla en la casilla de “Email address”.



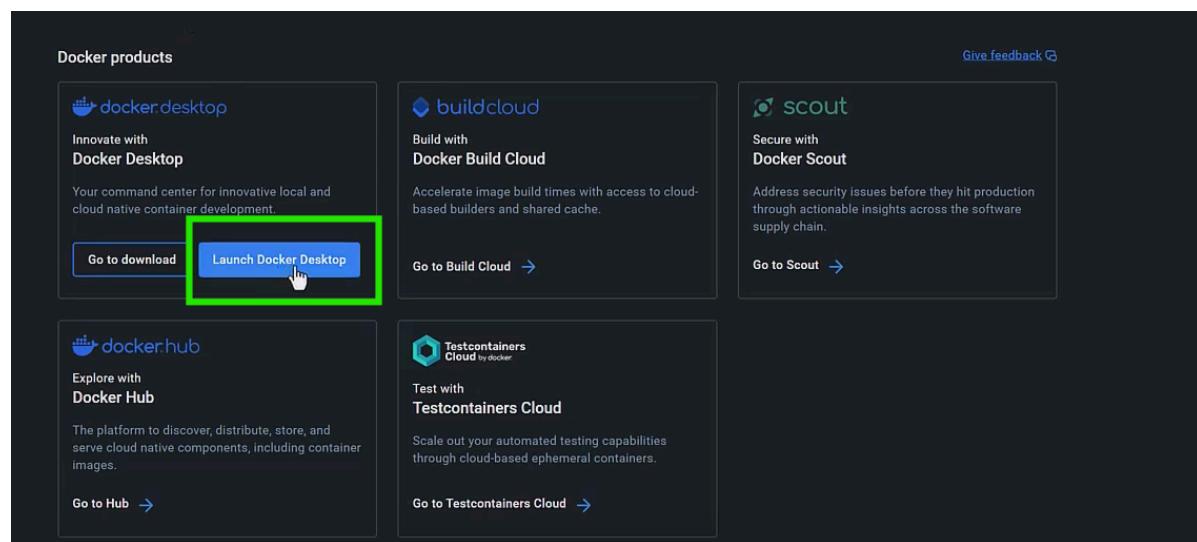
Si por el contrario no tenemos, deberemos darle a la casilla de “*Create an account*” para crearnos una cuenta nueva. Tenemos la opción de poder utilizar nuestra cuenta de *Google* o *Github*, si no queremos crear una cuenta desde cero.

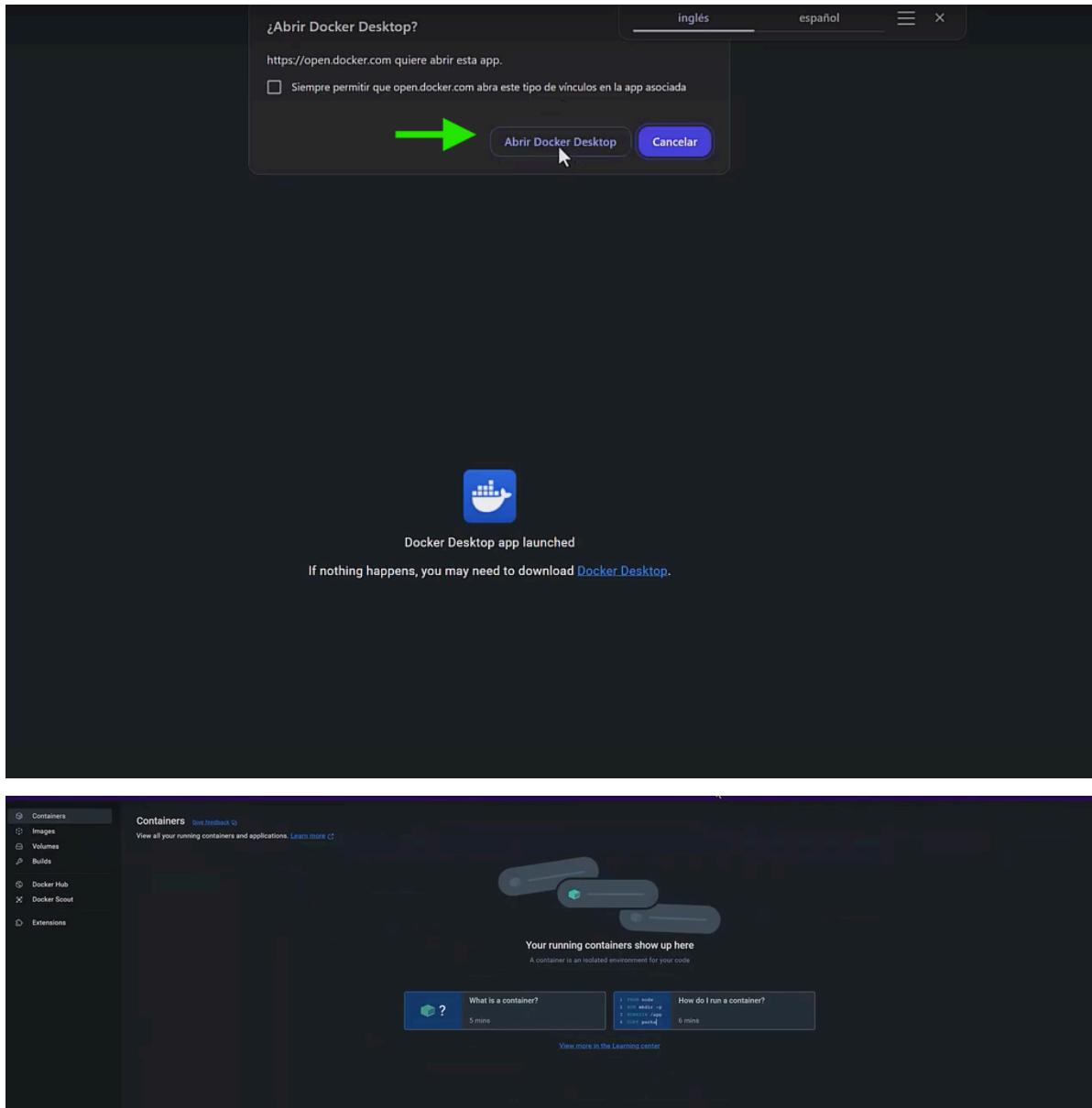


Pongamos que si tenemos una cuenta creada, pulsamos la tecla de *Sign in* que podemos observar en la imagen anterior y a continuación, nos saldrá la siguiente imagen:



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





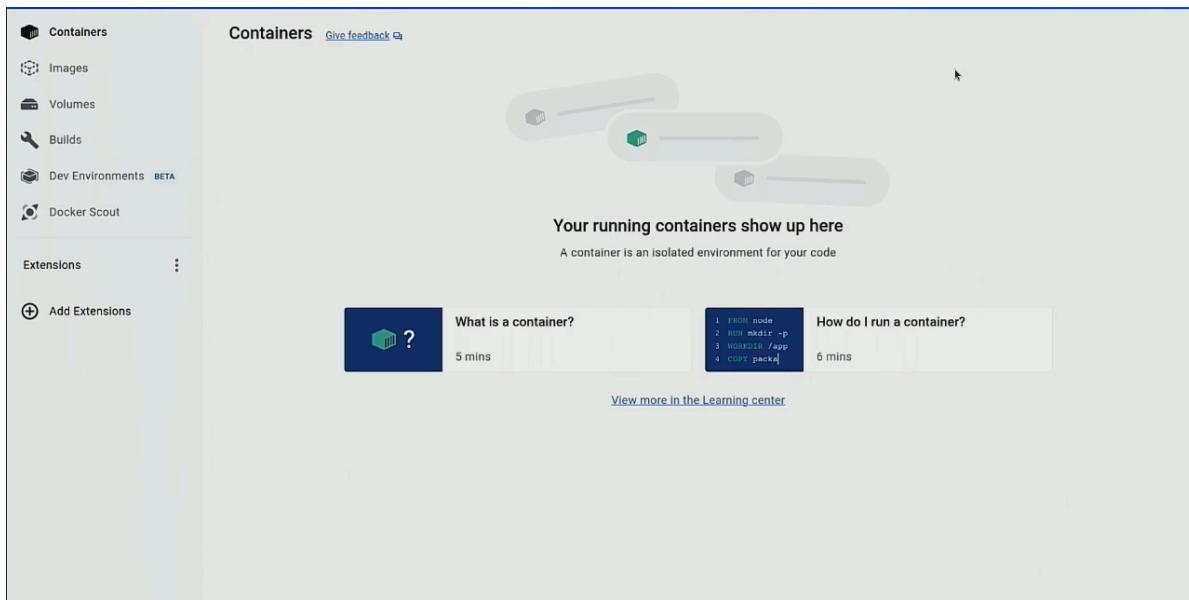
3. Interfaz de Docker Desktop

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.

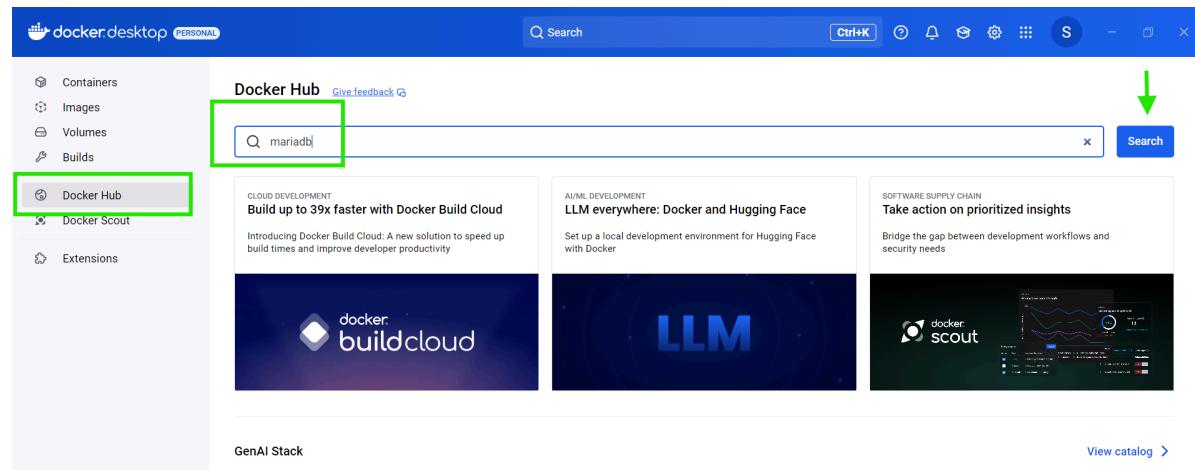


3.2. Images (Imágenes)

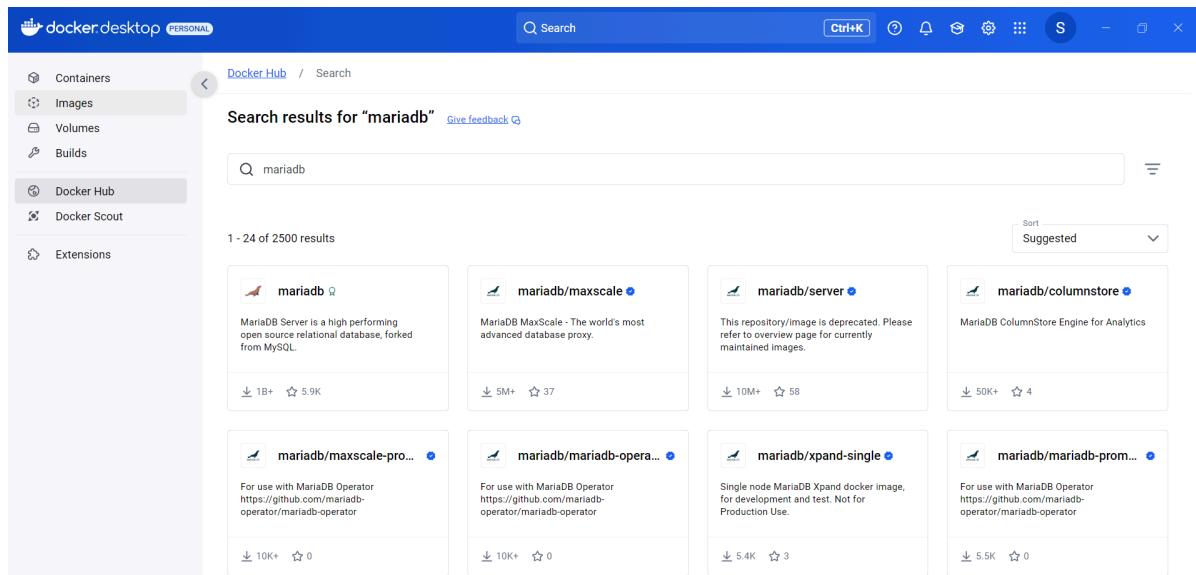
A través del área de *imágenes*, podemos gestionar todas las **imágenes Docker**. Las imágenes son plantillas para crear contenedores. Desde esta sección puedes:

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

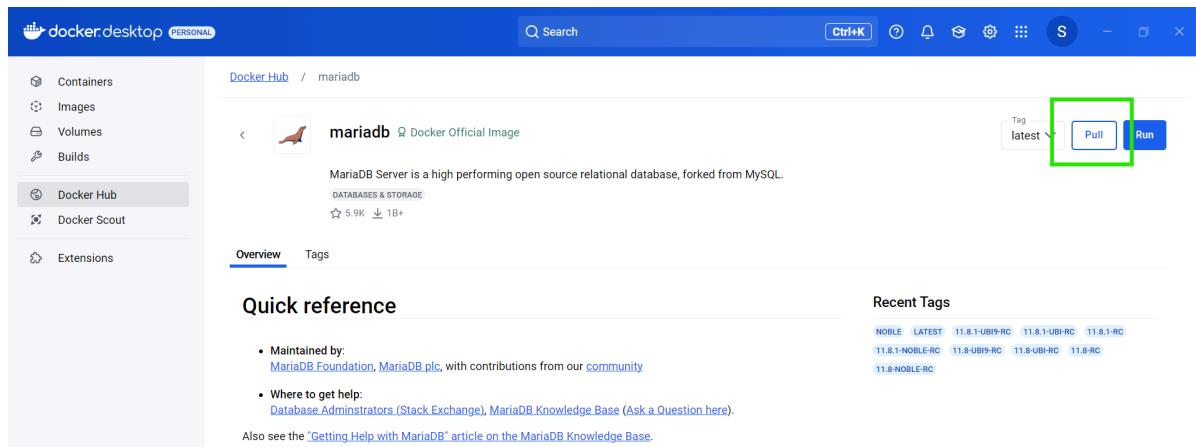
Ejemplo práctico: Vamos al área de Docker hub y buscamos la imagen que queramos descargar. En nuestro ejemplo es la última imagen de mariadb:



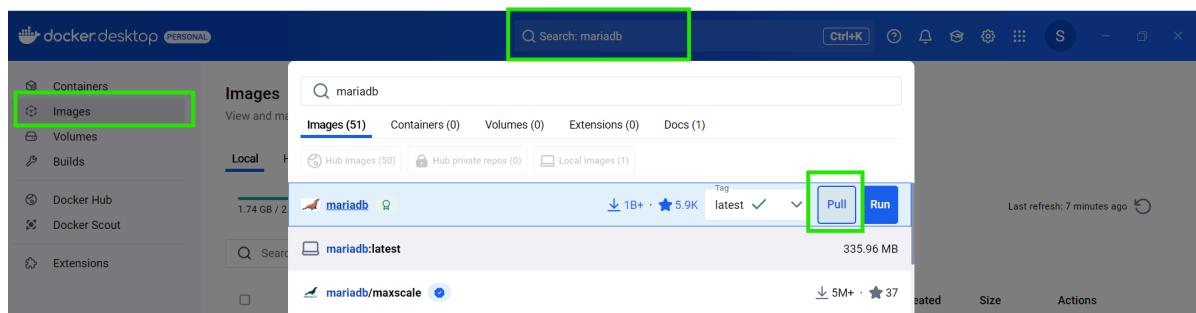
Nos aparecen las diferentes imágenes que existen con dicho nombre:



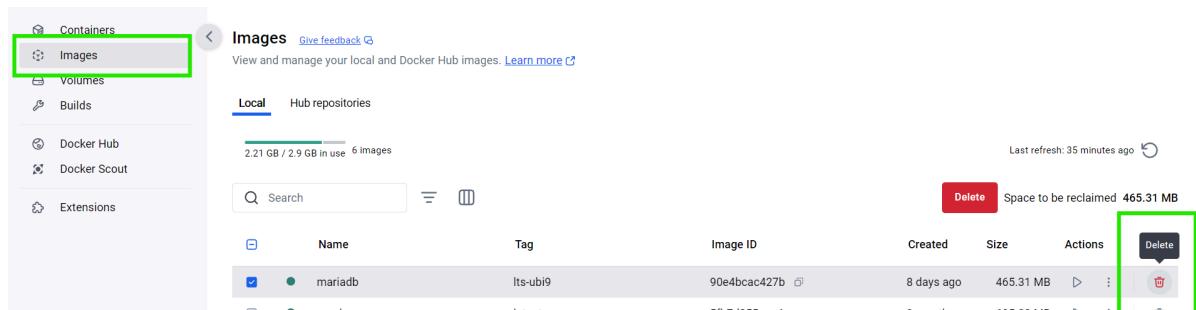
Seleccionamos la oficial y clickamos en *Pull* para descargarla:



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", "OS/ARCH", "Vulnerabilities" (with a severity matrix), and "Compressed Size".

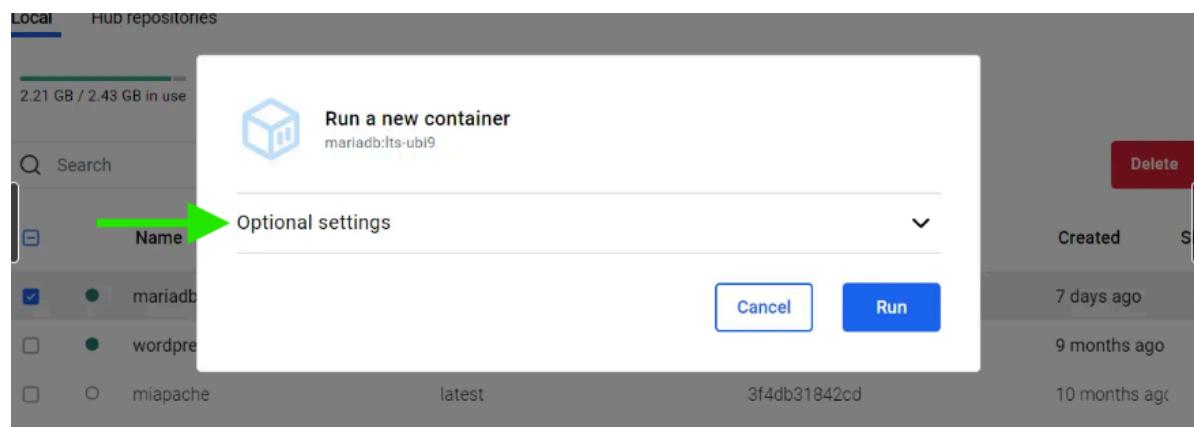
Digest	OS/ARCH	Vulnerabilities	Compressed Size
9586c56a4e03	linux/amd64	3 32 14 6 5	118.83 MB
5dde1f996e76	linux/arm64	3 32 14 6 5	116.79 MB
89e47d5278dd	linux/ppc64le	3 32 14 6 5	129.33 MB
f78583006cc9	linux/s390x	3 32 14 6 5	124.99 MB

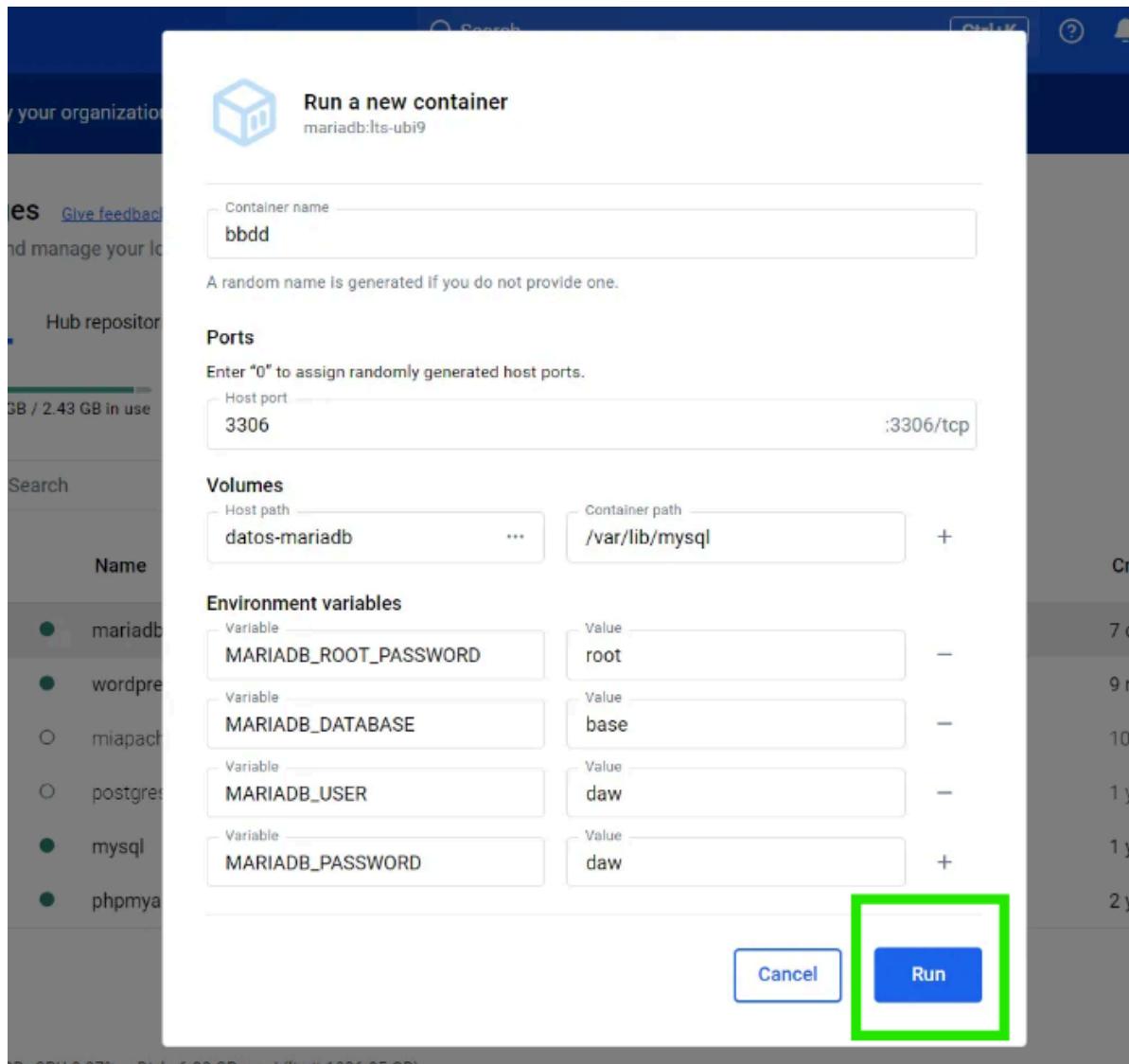
3.3. Containers (Contenedores)

Desde aquí puedes:

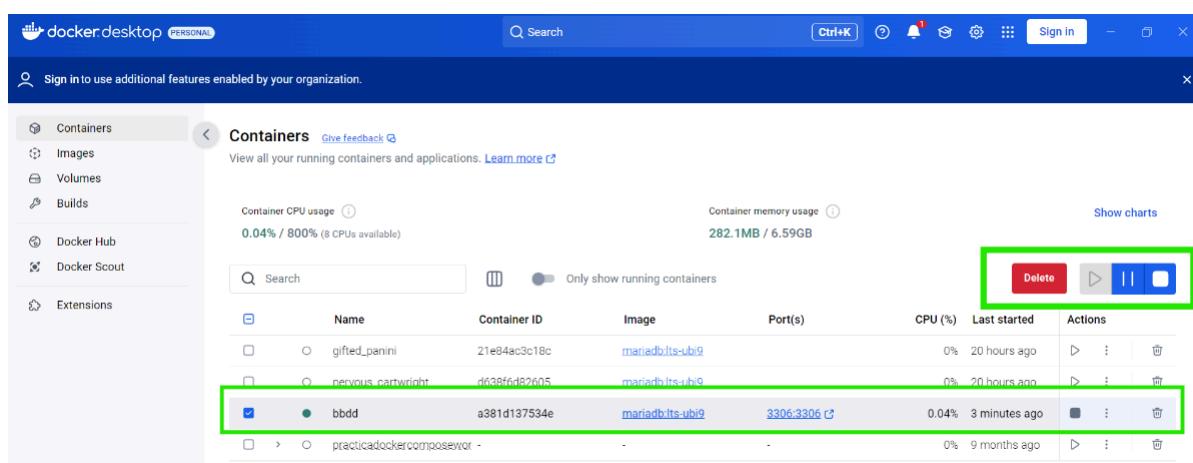
- **Gestionar contenedores:** Configurar, detener, eliminar, pausar y reiniciar contenedores. Puedes interactuar con los contenedores sin tener que usar la terminal.

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

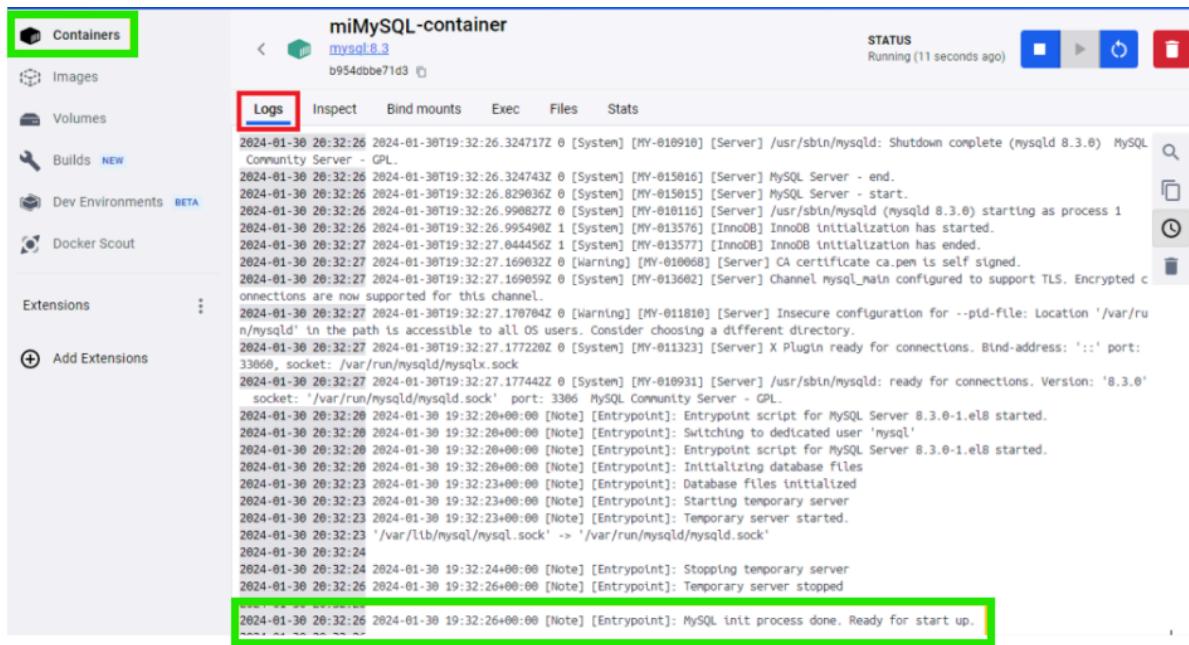




Una vez introducidos los datos, pulsamos en *RUN* para su creación. En el área de "containers" te aparecerá lo siguiente:



- **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 selected. A volume named 'datos-mariadb' is highlighted with a green box. The table lists several volumes, including their names, creation times, sizes, and actions (Edit and Delete).

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	
a70d467000b0eb70ffbd01e001e9e55d0b411e4e4727-20bb423c76d0e0	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.

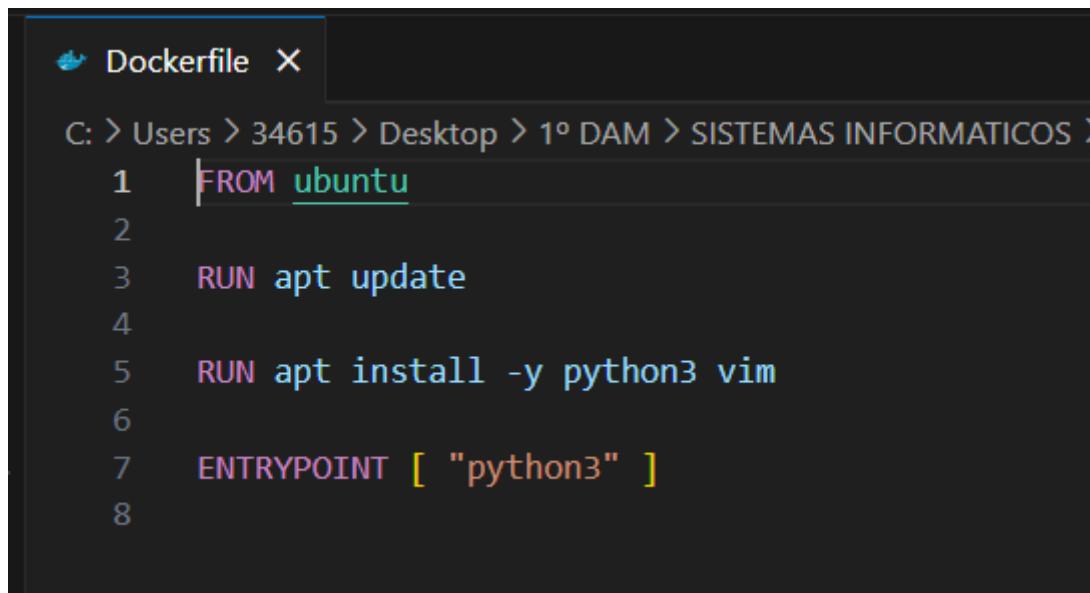
Pero, ¿qué es un **Dockerfile**?

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**.

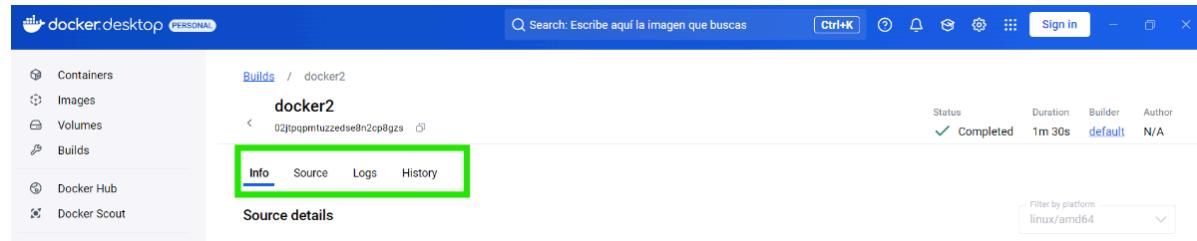
Si queremos ejecutar un script Python al iniciar el contenedor, deberemos ejecutar el siguiente comando:

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

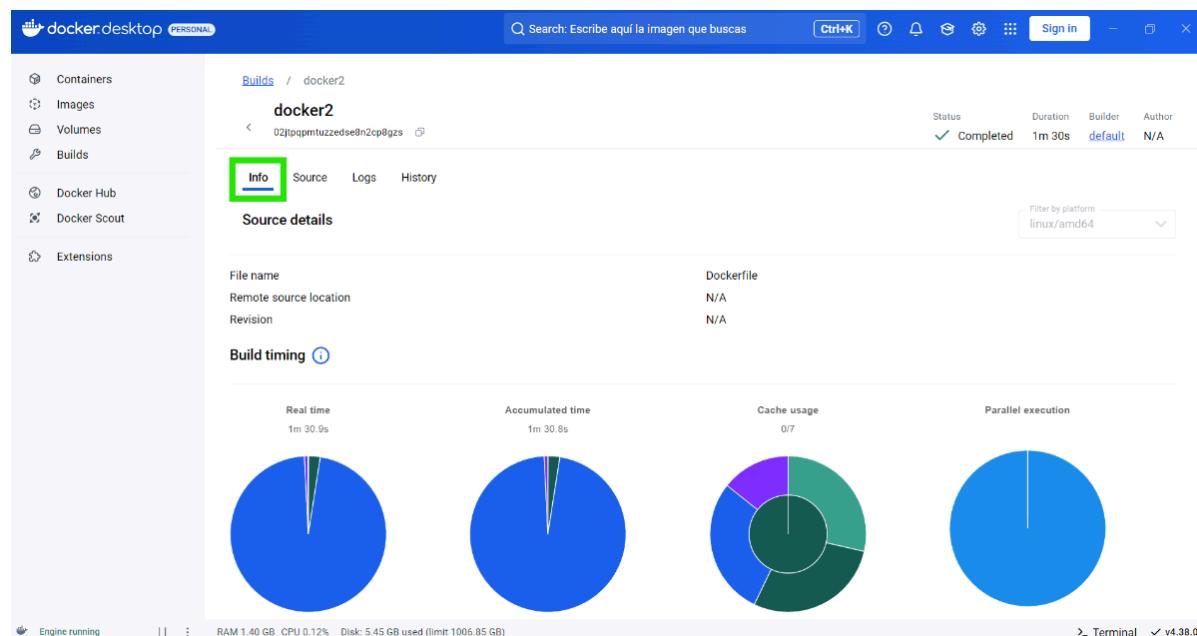
Y ejecutará `python3 script.py` automáticamente.

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 it, "docker2" is listed with a status of "Completed" (green checkmark), duration "1m 30s", builder "default", and author "N/A". There are tabs for "Info", "Source" (which is selected and highlighted in green), "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 build "docker2" is shown again with its status as "Completed". The logs output is as follows:

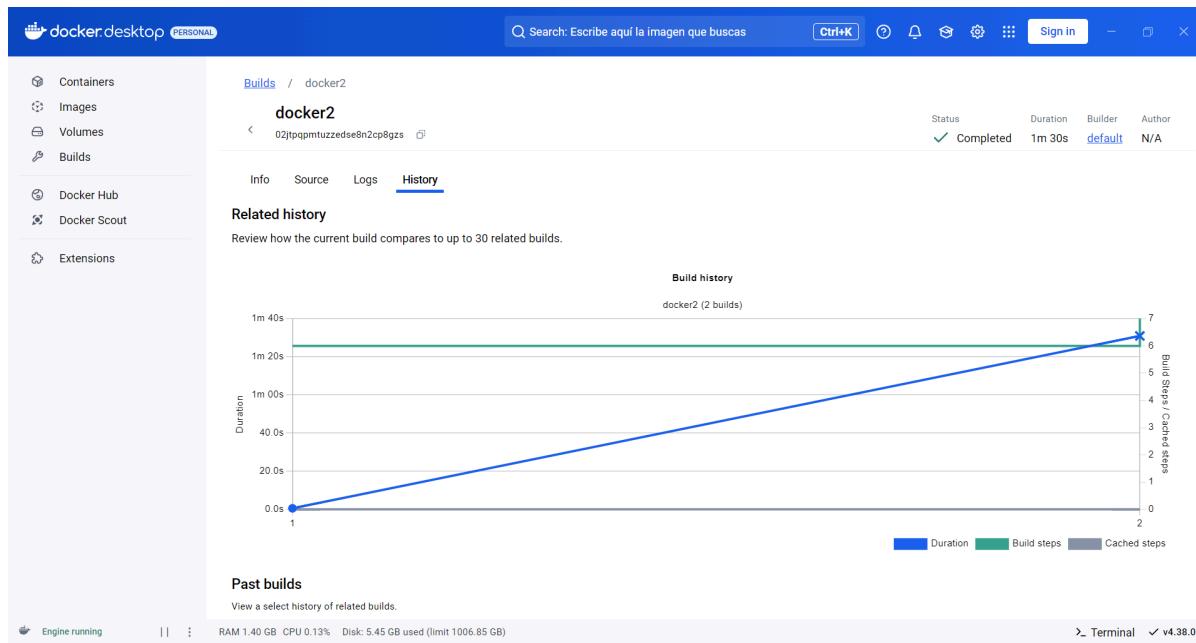
- ✓ INTERNAL load build definition from Dockerfile transferring 0.0s 0.0s
- ✓ INTERNAL load metadata for docker.io/library/ubuntu:latest 2.1s
- ✓ INTERNAL load .dockerignore transferring 0.0s 0.0s
- ✓ 1/3 FROM docker.io/library/ubuntu:latest@sha256:1b8d8ff4777f36f19bfe73ee4df61e3a0b789caeff29caa019539ec7c9a57f95 0.0s
 - resolve docker.io/library/ubuntu:latest@sha256:1b8d8ff4777f36f19bfe73ee4df61e3a0b789caeff29caa019539ec7c9a57f95 0.0s
 - done 1.1 kB -1.-7s
 - done 424 Bytes -1.-5s
 - done 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.

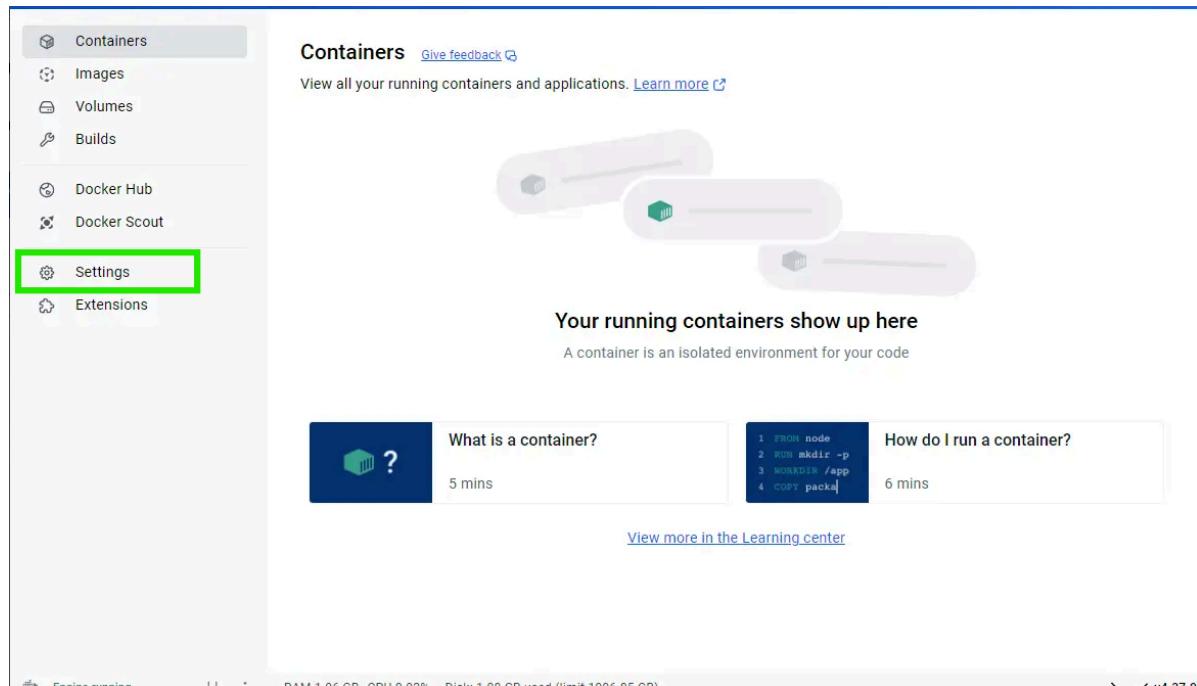
3.7. Extensions (Extensões)

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.

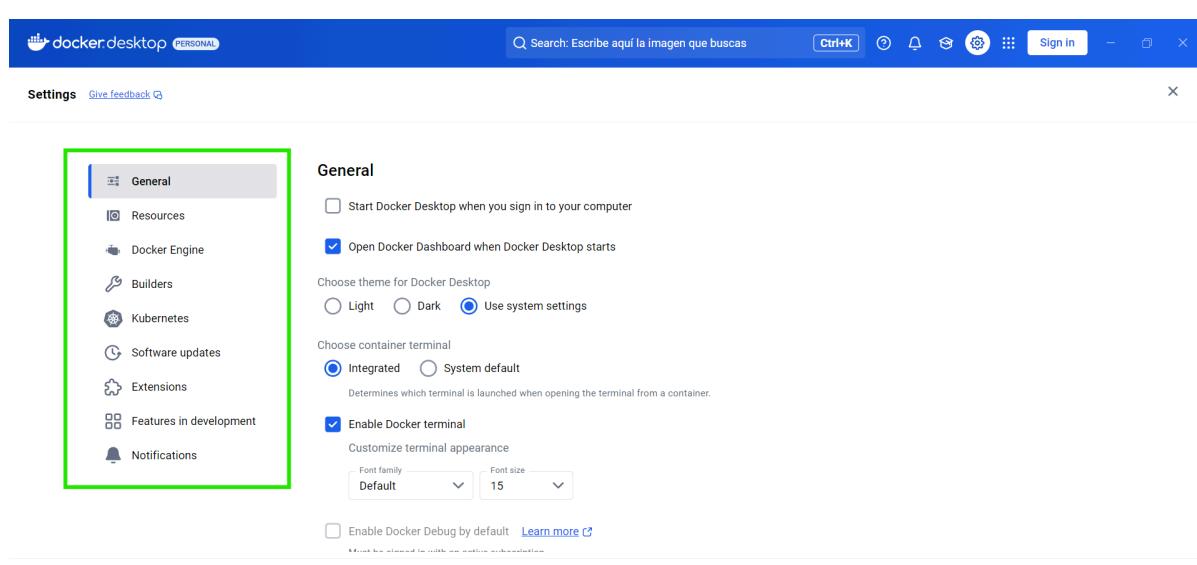
Estas extensiones pueden ayudarte a realizar **tareas específicas**, como:

- 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)



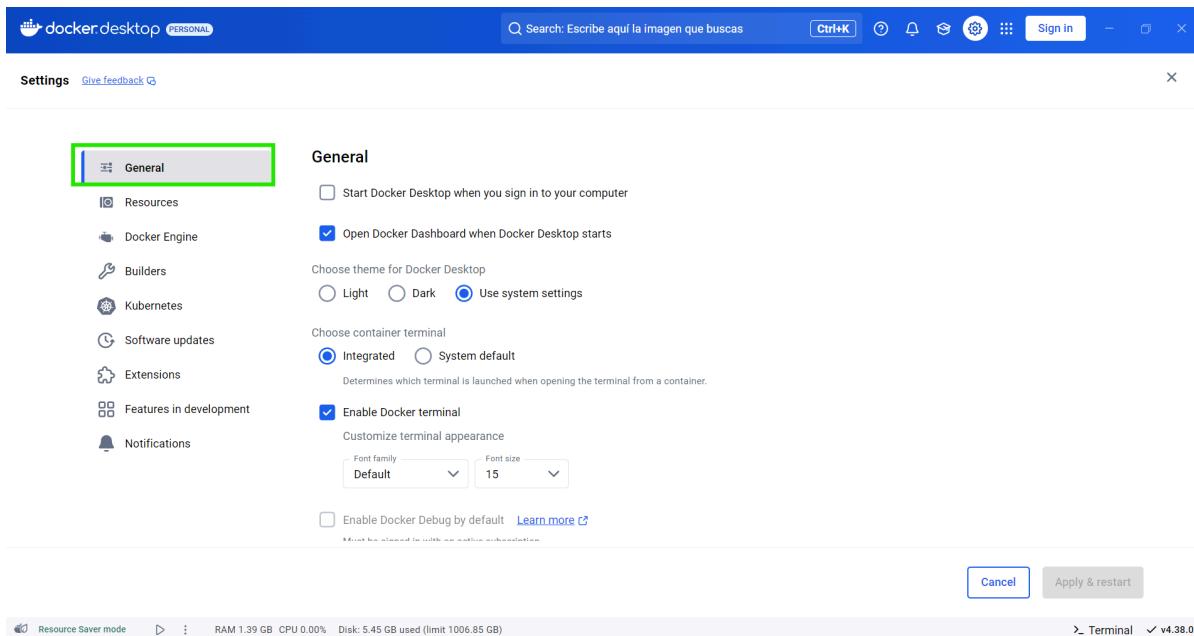
Aquí puedes ajustar varias configuraciones de Docker Desktop, tales como:



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:

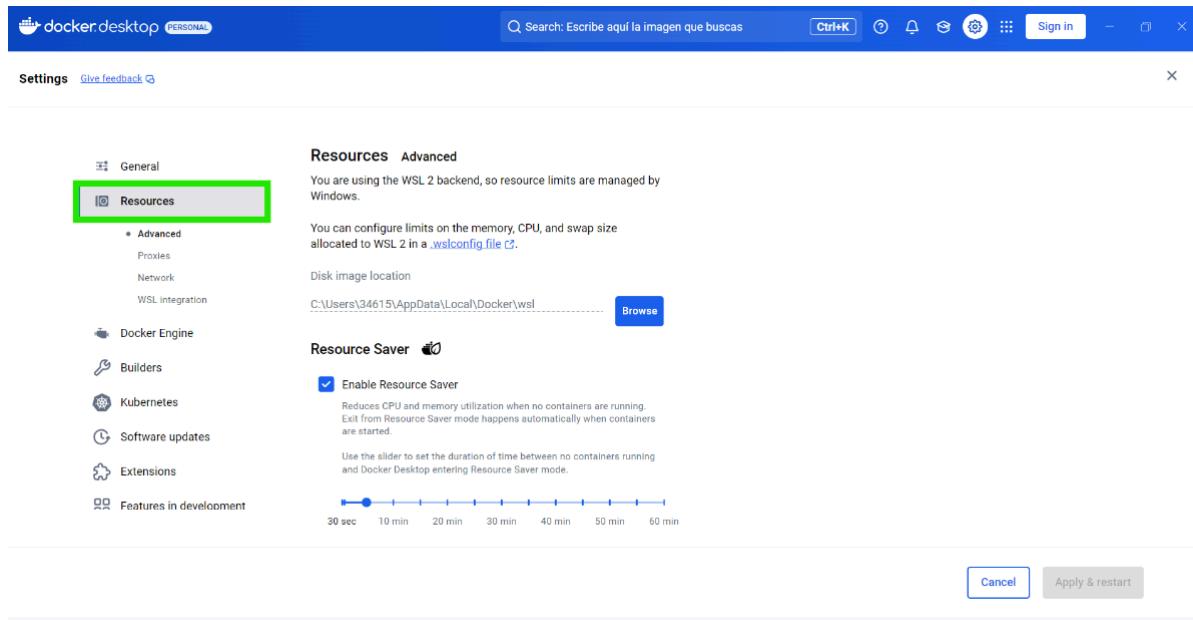
Aquí puedes configurar un **proxy** si tu red requiere uno para acceder a internet. Esto es común en entornos corporativos o redes restringidas, donde Docker necesita acceder a recursos externos como **Docker Hub** a través de un proxy.

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?

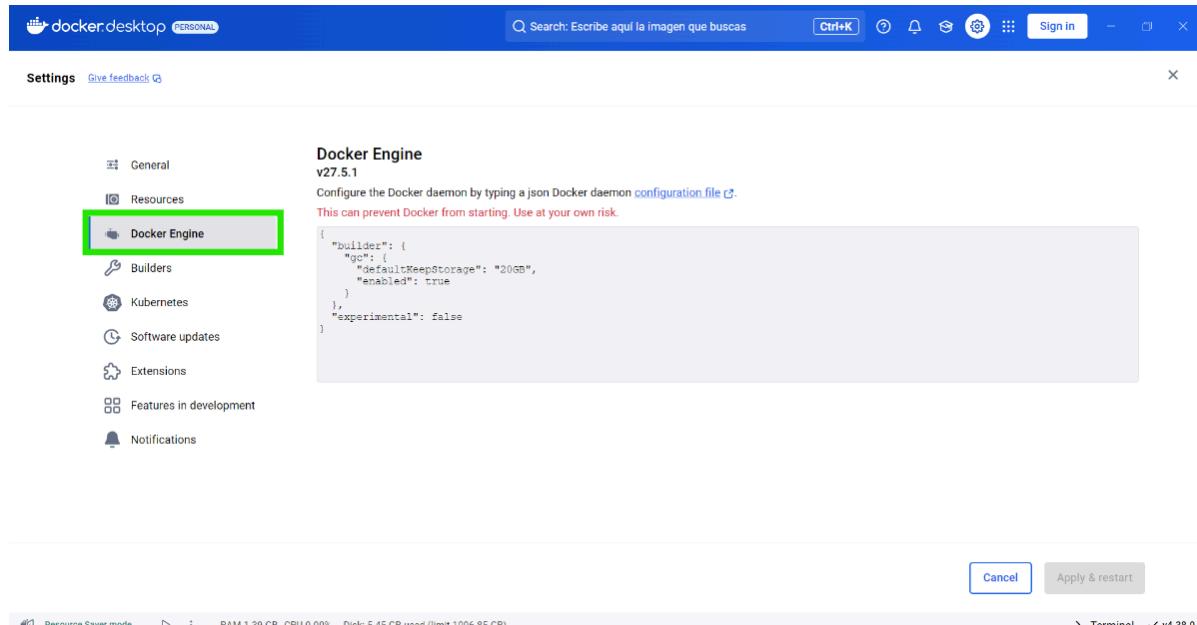
En la sección **Resources**, puedes configurar la cantidad de recursos del sistema que Docker usará, como la CPU, la memoria RAM y el espacio en disco. Esto es especialmente importante cuando trabajas con contenedores que requieren más recursos o si tienes un sistema con capacidades limitadas.

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?

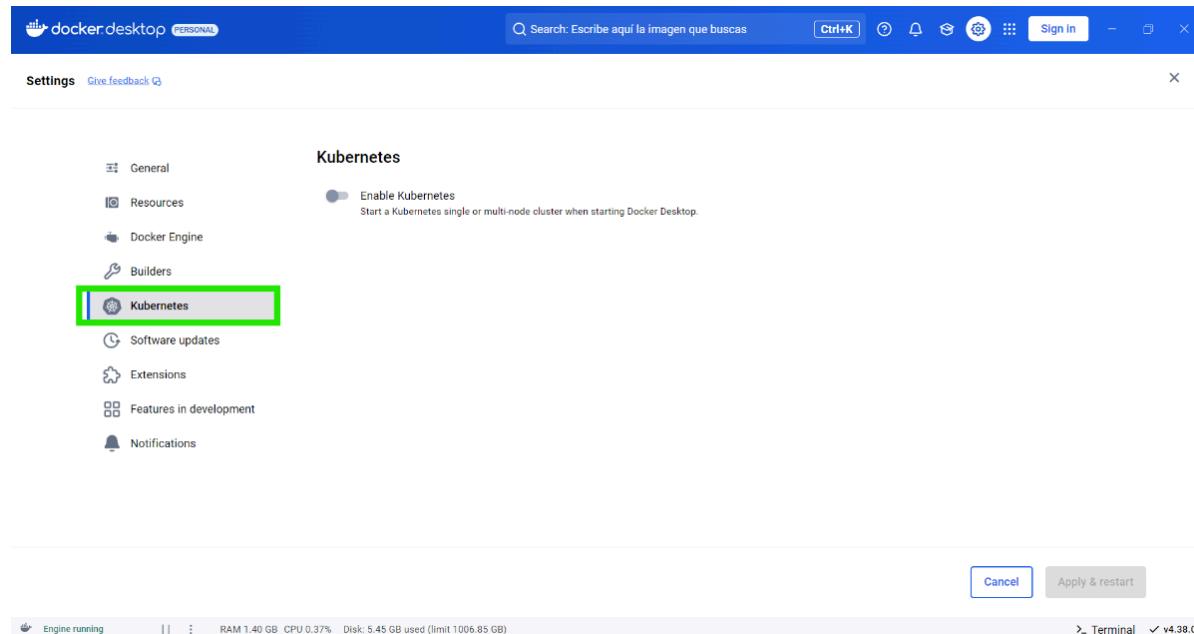
En la sección **Docker Engine**, puedes configurar directamente los parámetros del motor Docker utilizando un archivo de configuración JSON. Esto te permite personalizar aún más la forma en que Docker opera en tu máquina.

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



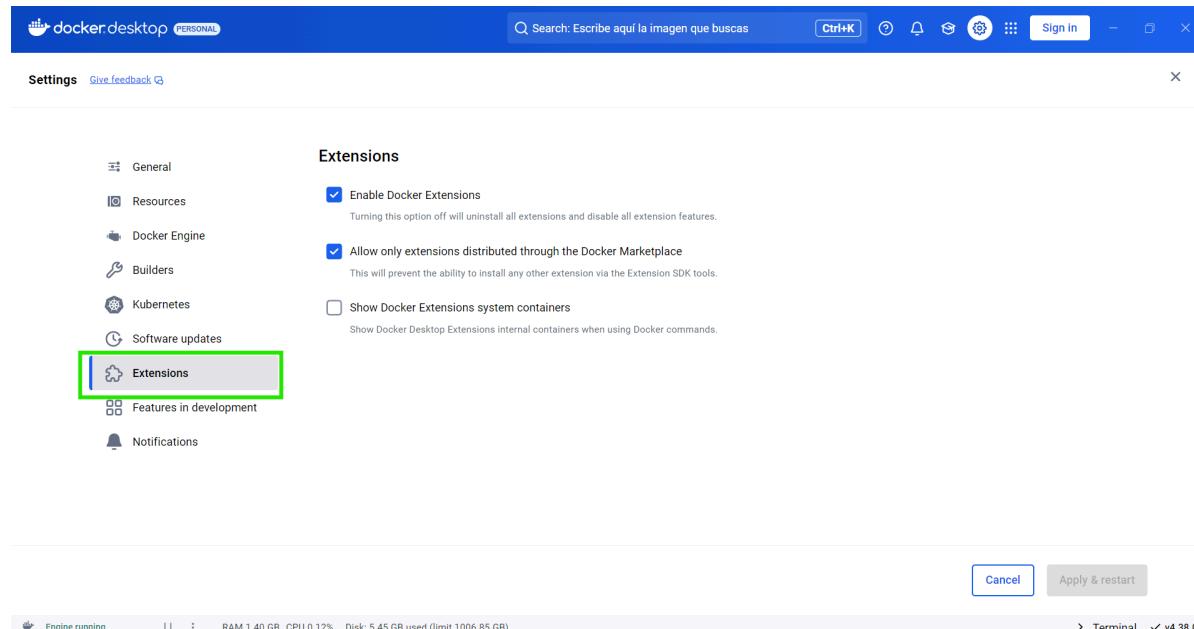
¿Para qué sirve?

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)



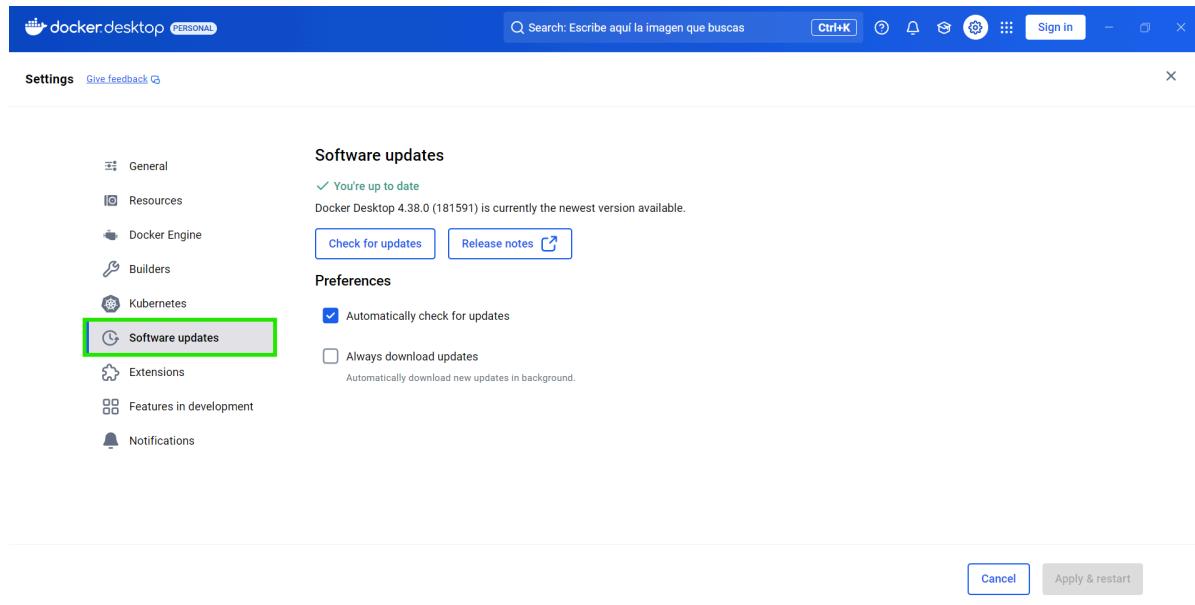
¿Para qué sirve?

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)



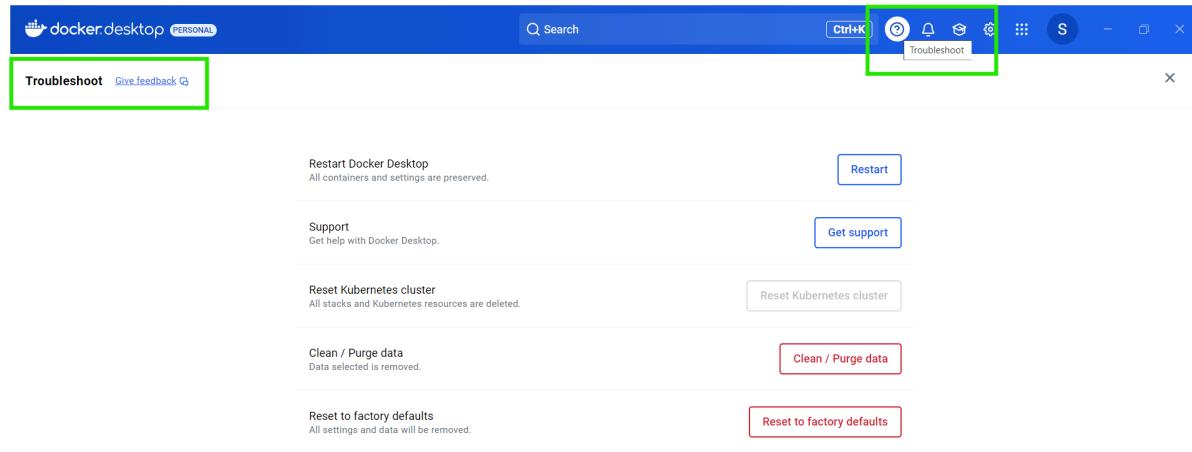
¿Para qué sirve?

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)



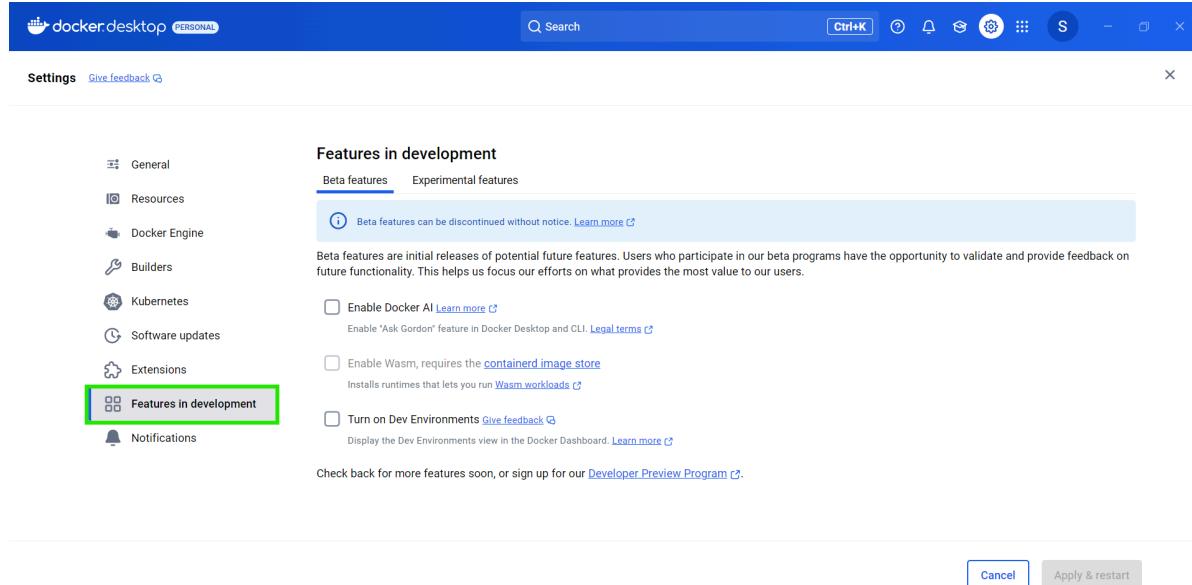
¿Para qué sirve?

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)



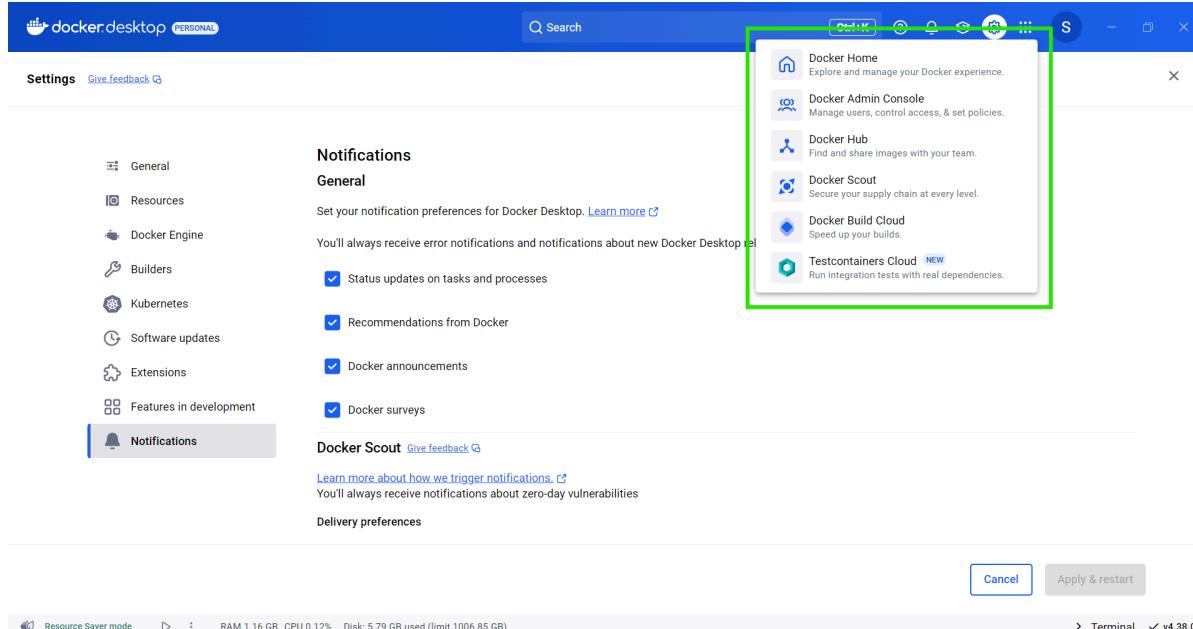
¿Para qué sirve?

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

Funcionalidad:

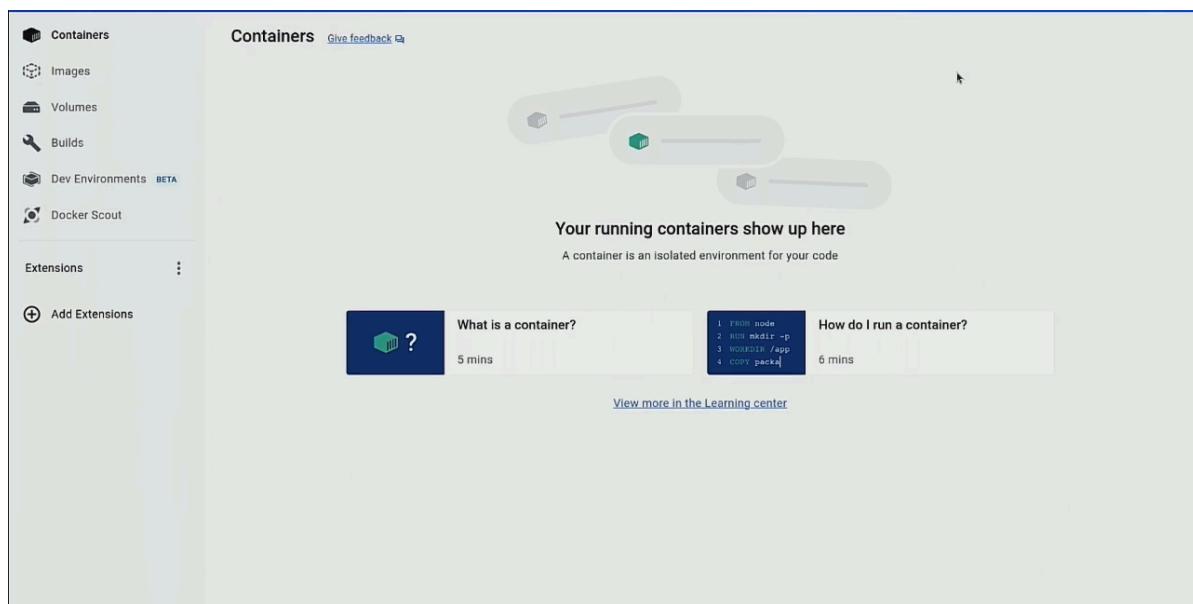
- **Activar características beta:** Algunas funciones de Docker están en versión beta o experimental. Si activas esta opción, puedes probar características nuevas que aún no están completamente disponibles para todos los usuarios.
- **Personalización avanzada:** Las características experimentales pueden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

4. Otras áreas y secciones:



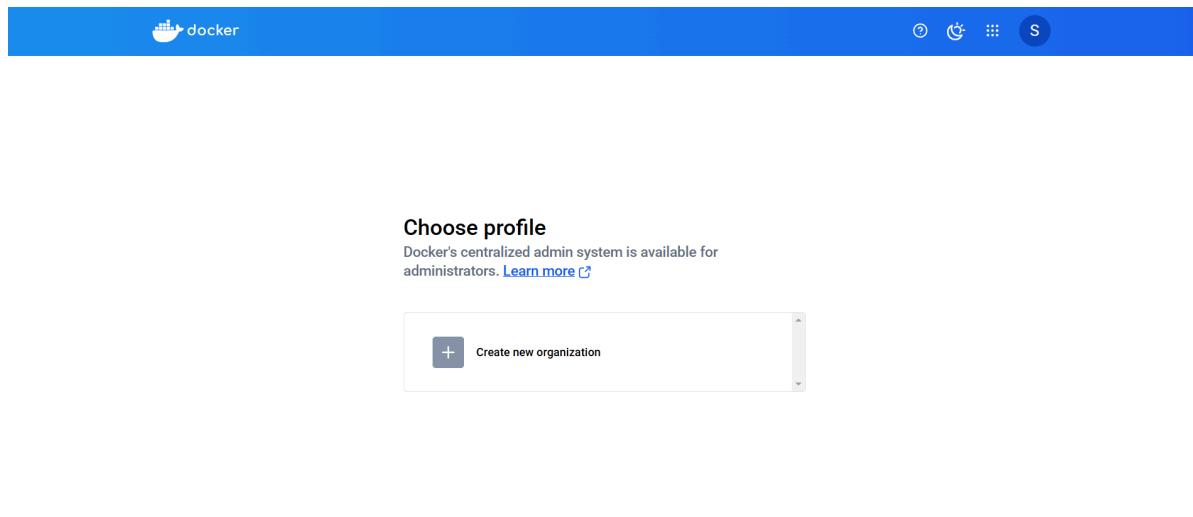
4.1. Docker Home:

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 central image of a white cloud with a small computer monitor icon inside it, showing arrows pointing in and out. Underneath this image, the text 'Powerful builders, in the cloud' is displayed. Below the text is a brief description: 'Docker Build Cloud significantly reduces build times, both locally and in CI, by providing a dedicated remote builder and shared cache. Powered by the cloud, developer time and local resources are freed up so your team can focus on more important things, like innovation.' At the bottom of the page are navigation buttons: 'Back', a series of three dots indicating steps, 'Skip', and a prominent '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. At the top, there's a header with the 'Testcontainers Cloud' logo, a user profile for 'Sandra Ruja...', and a 'Skip' button. Below the header is a purple banner with the text '50 Cloud Runtime minutes' and a 'Personal' section. A dropdown menu for 'Sandra Rujas Arroyo' is open. To the right of the banner is a progress bar with two steps: 'VERIFY' and 'NEXT STEPS'. The main content area has tabs for 'Desktop' (selected) and 'Continuous Integration (CI)'. Under 'Desktop', there are links for 'Mac OS', 'Windows', 'Ubuntu', and 'Fedora'. To the right, there's a section titled 'Use Testcontainers Desktop on Windows' with two numbered steps: '1. Download the latest version of Testcontainers Desktop' (with a 'Download' button) and '2. Install the client' (with a screenshot of the 'Testcontainers Desktop Setup' wizard). To the far right, a box says 'Waiting for client...' with the instruction 'Install and run Testcontainers Cloud on your computer to continue.'

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.