

docker®

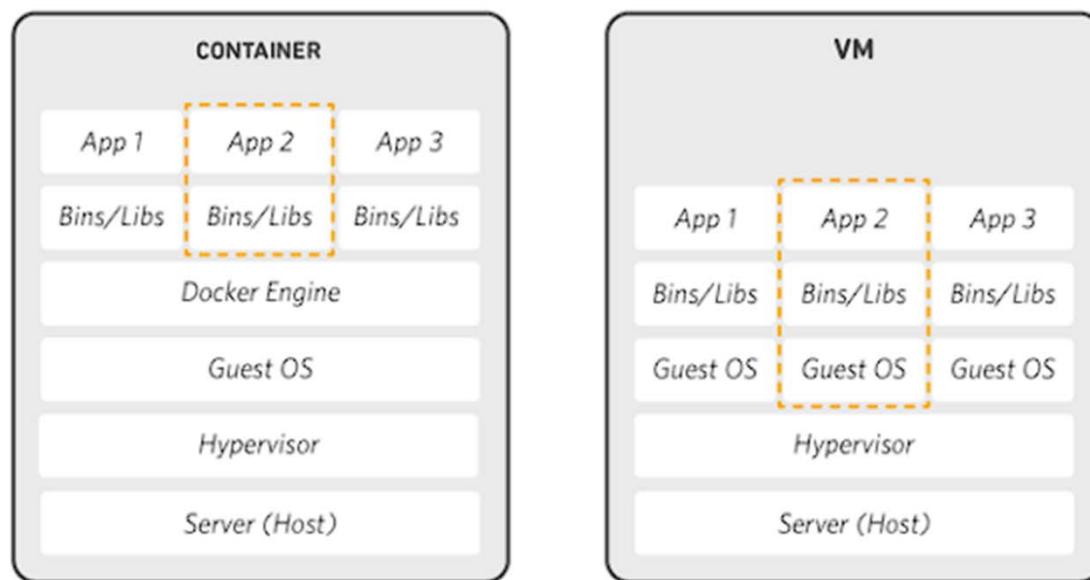
Introducción

- Docker es una plataforma de software que permite crear, probar e implementar aplicaciones rápidamente. Docker empaqueta software en unidades estandarizadas llamadas contenedores que incluyen todo lo necesario para que el software se ejecute, incluidas bibliotecas, herramientas de sistema, código y tiempo de ejecución.

Introducción

- Con Docker, puede implementar y ajustar la escala de aplicaciones rápidamente en cualquier entorno con la certeza de saber que su código se ejecutará.
- Docker es un sistema operativo para contenedores. De manera similar a cómo una máquina virtual virtualiza (elimina la necesidad de administrar directamente) el hardware del servidor, los contenedores virtualizan el sistema operativo de un servidor.

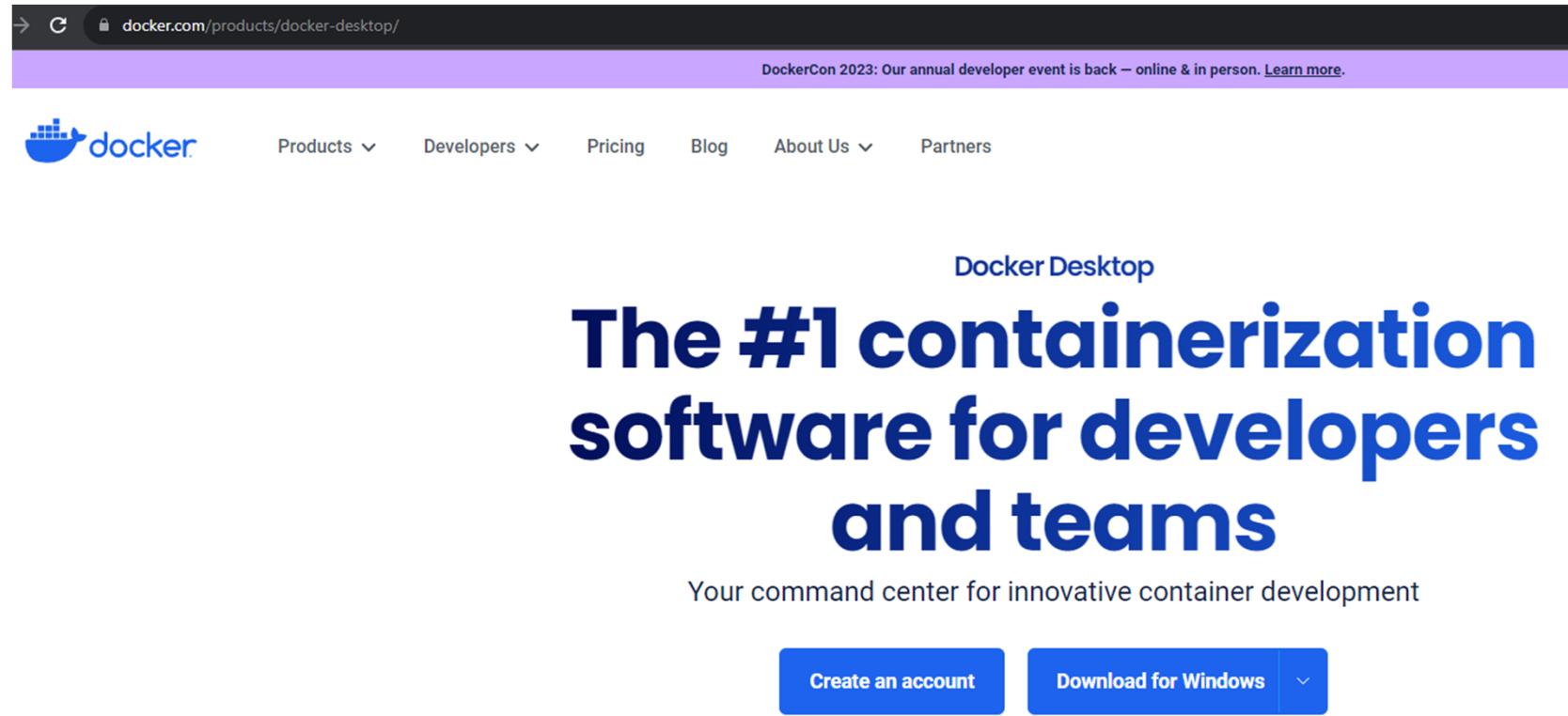
Introducción



Introducción

- Los contenedores permiten que varios componentes de cierta aplicación comparten los recursos de una única instancia del sistema operativo anfitrión. Este uso compartido es similar a la forma en que un hipervisor permite que varias máquinas virtuales (VM) comparten la unidad central de procesamiento (CPU) de un único servidor de hardware, la memoria y otros recursos.

Docker Desktop



The screenshot shows the Docker Desktop product page on the official Docker website. At the top, there's a navigation bar with links for Products, Developers, Pricing, Blog, About Us, and Partners. A purple banner at the top of the main content area reads "DockerCon 2023: Our annual developer event is back – online & in person. [Learn more.](#)". The main headline is "The #1 containerization software for developers and teams". Below it, a sub-headline says "Your command center for innovative container development". There are two prominent blue buttons: "Create an account" and "Download for Windows".

docker.com/products/docker-desktop/

DockerCon 2023: Our annual developer event is back – online & in person. [Learn more.](#)

Products ▾ Developers ▾ Pricing Blog About Us ▾ Partners

Docker Desktop

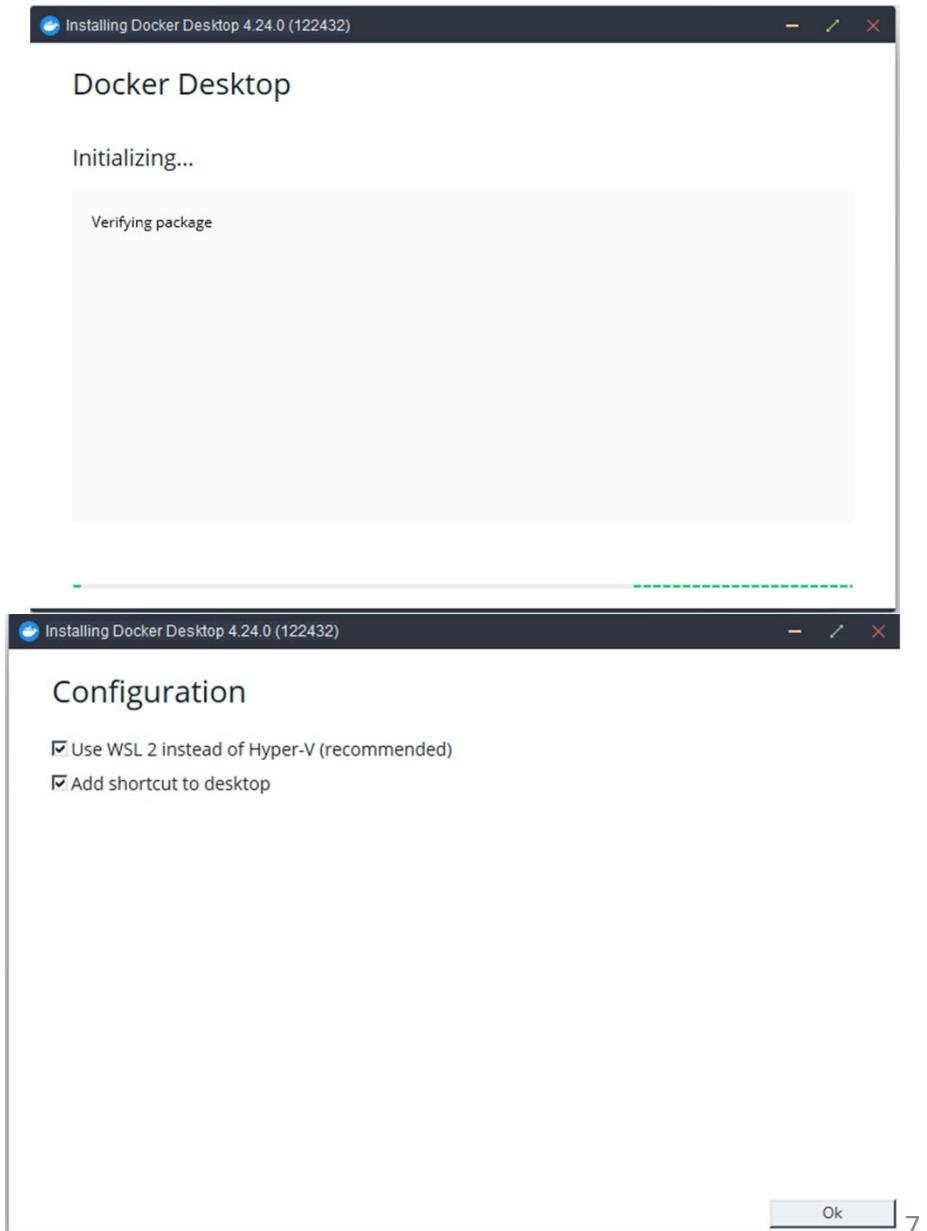
The #1 containerization software for developers and teams

Your command center for innovative container development

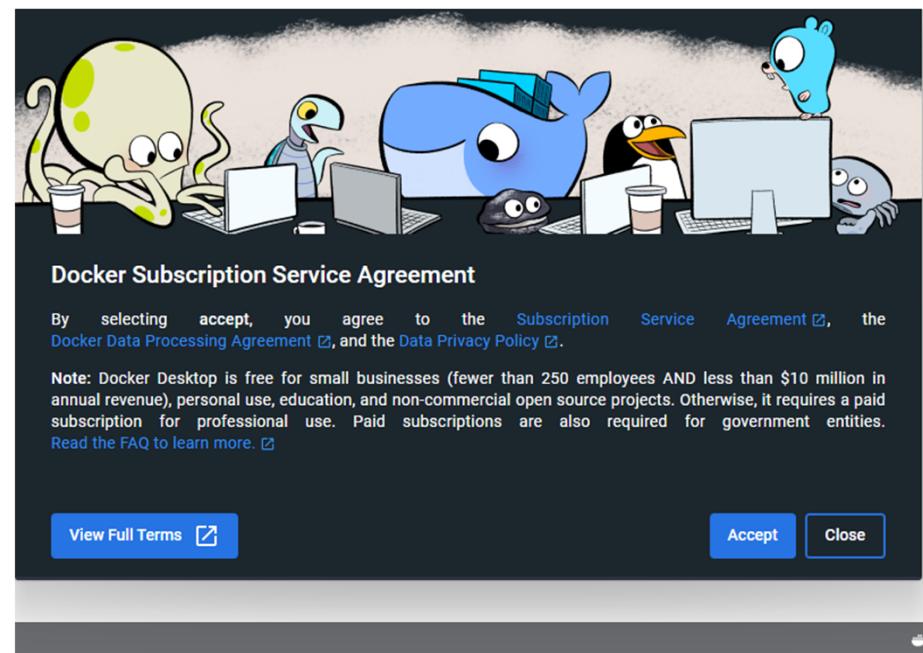
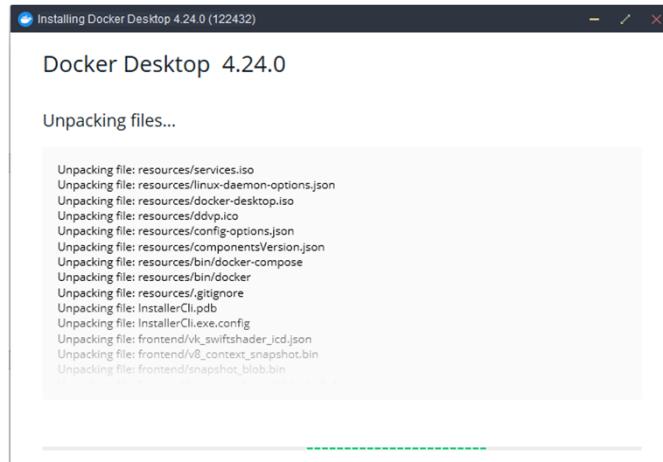
Create an account

Download for Windows ▾

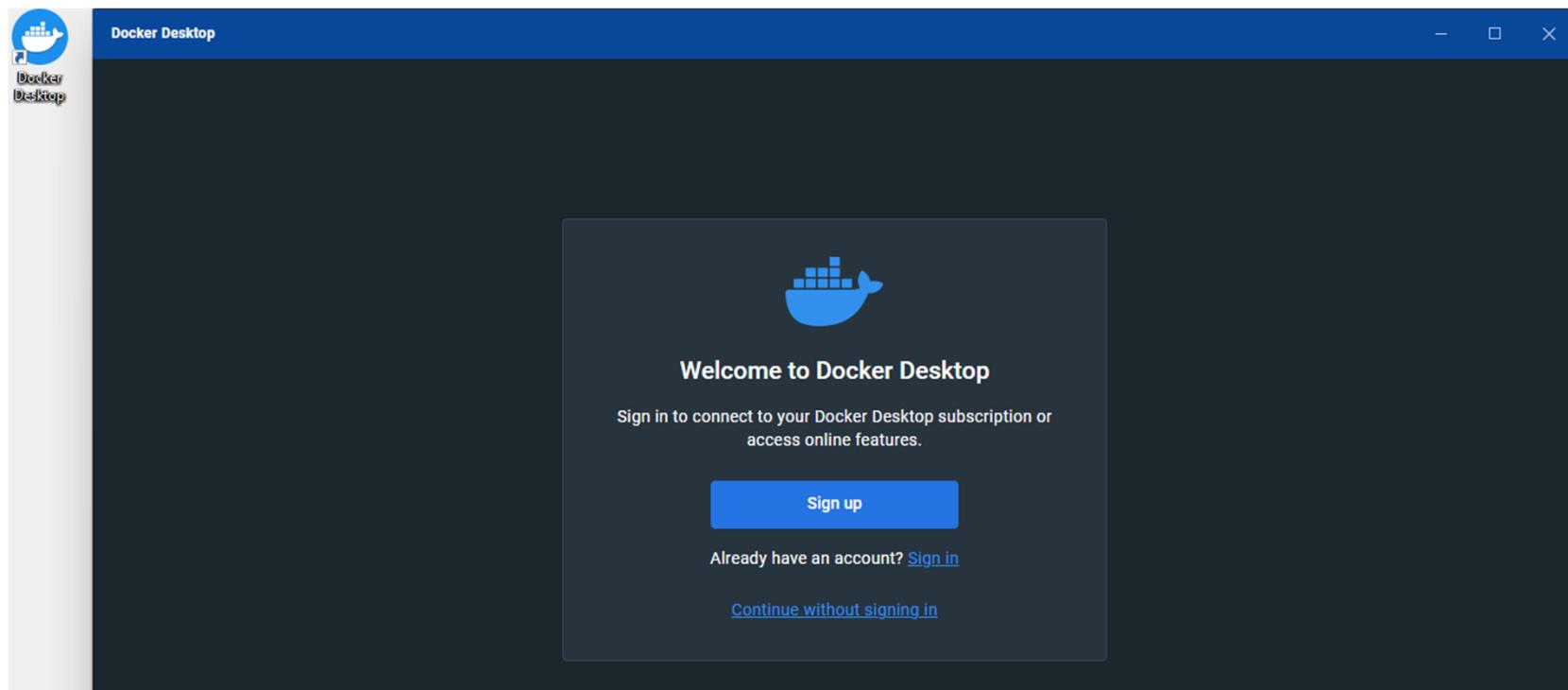
El instalador



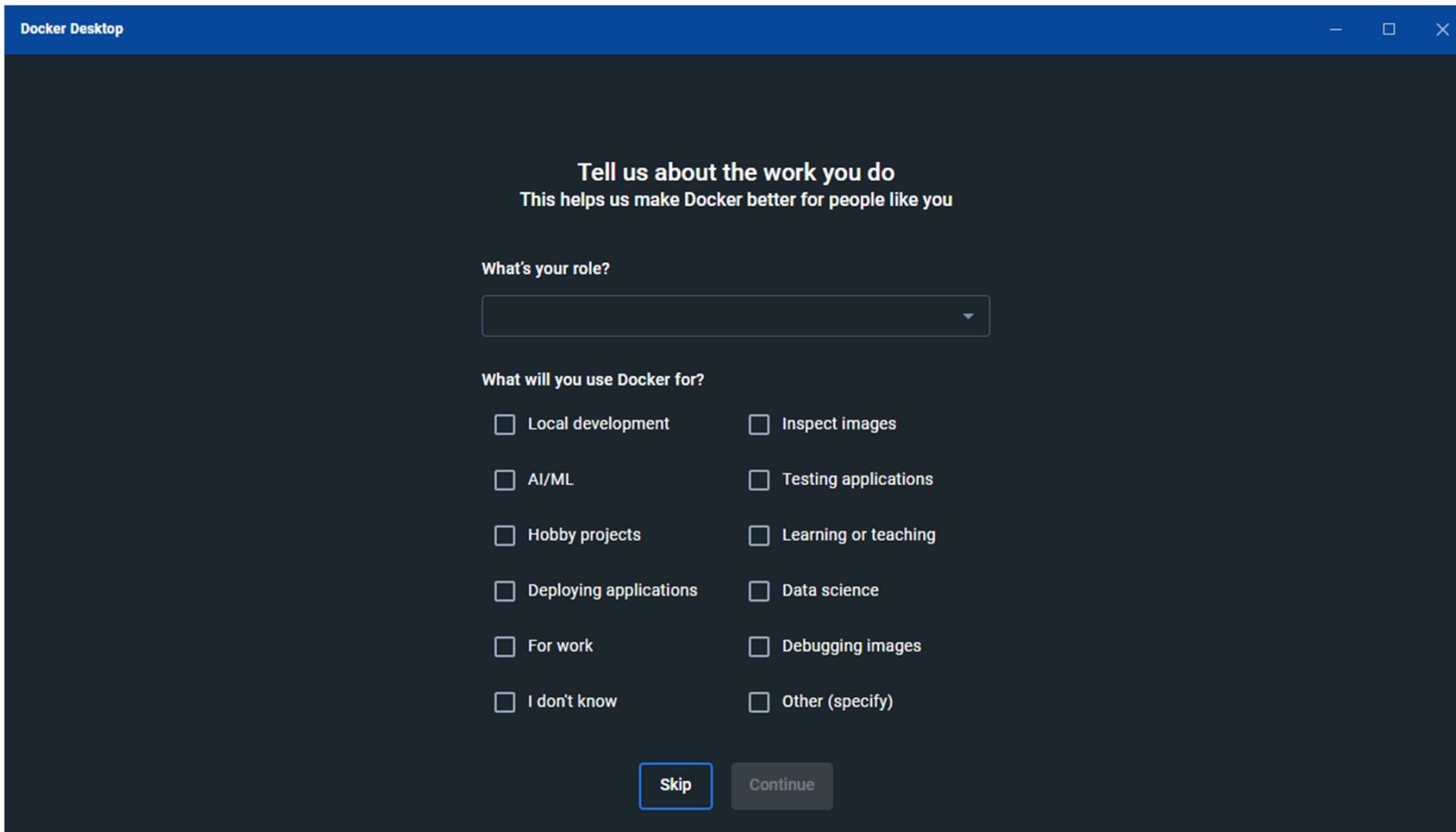
Instalando ...



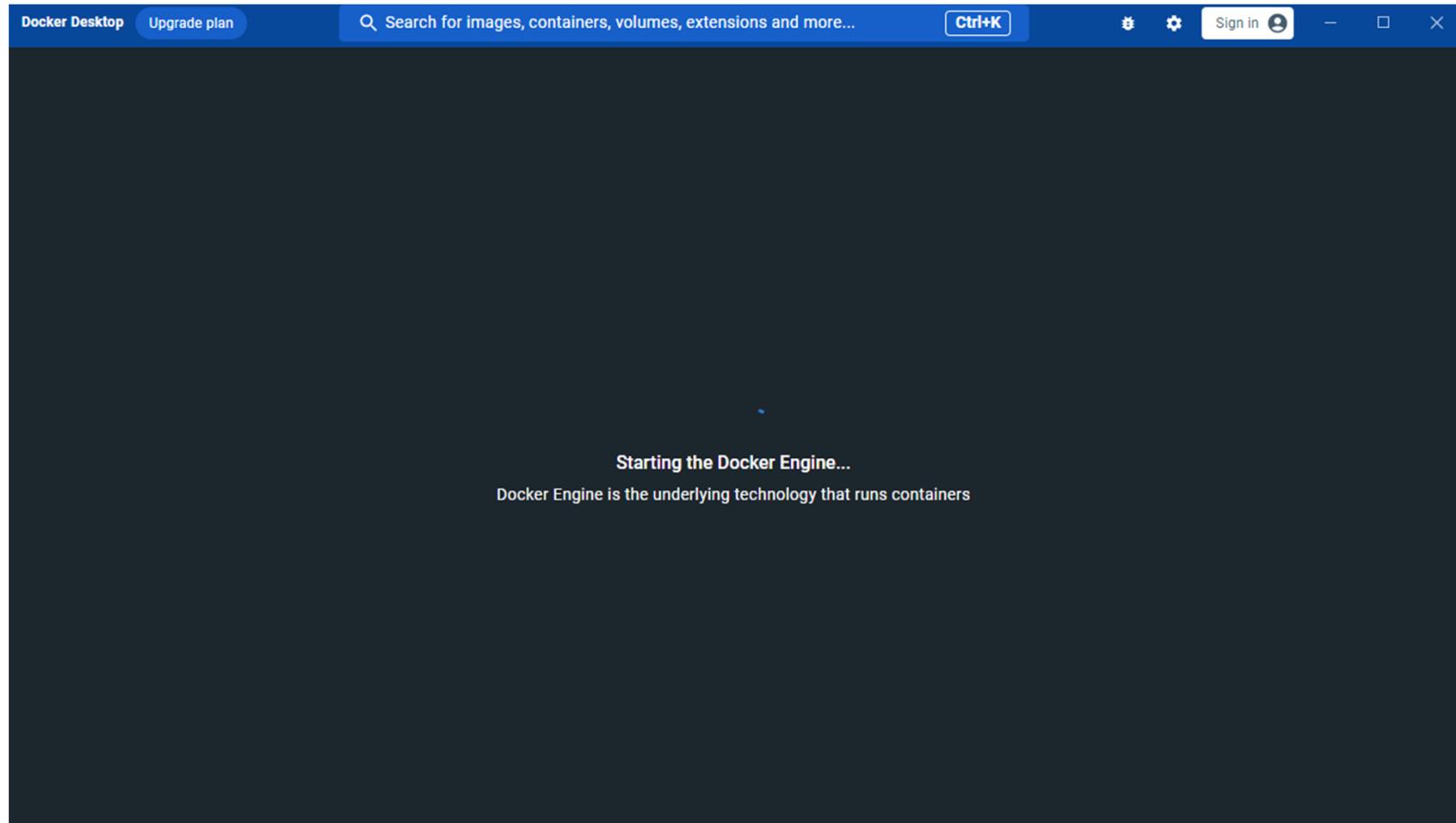
Instalado ¿?



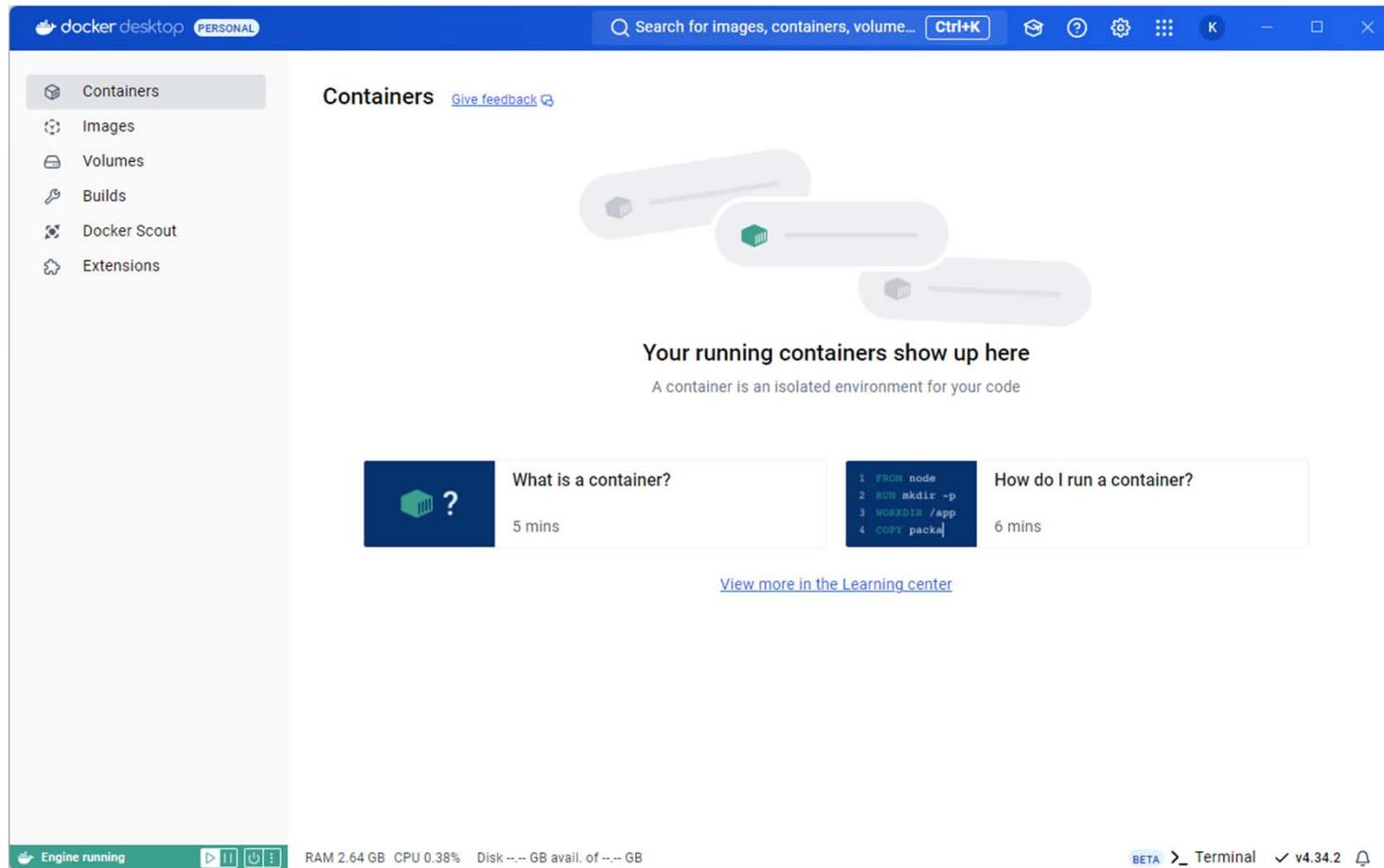
Más preguntas



Ahora parece que ya inicio

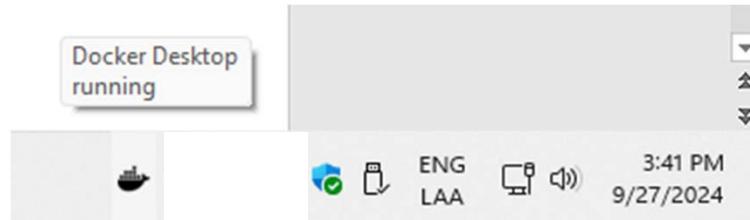


Según ya esta ejecutándose ...

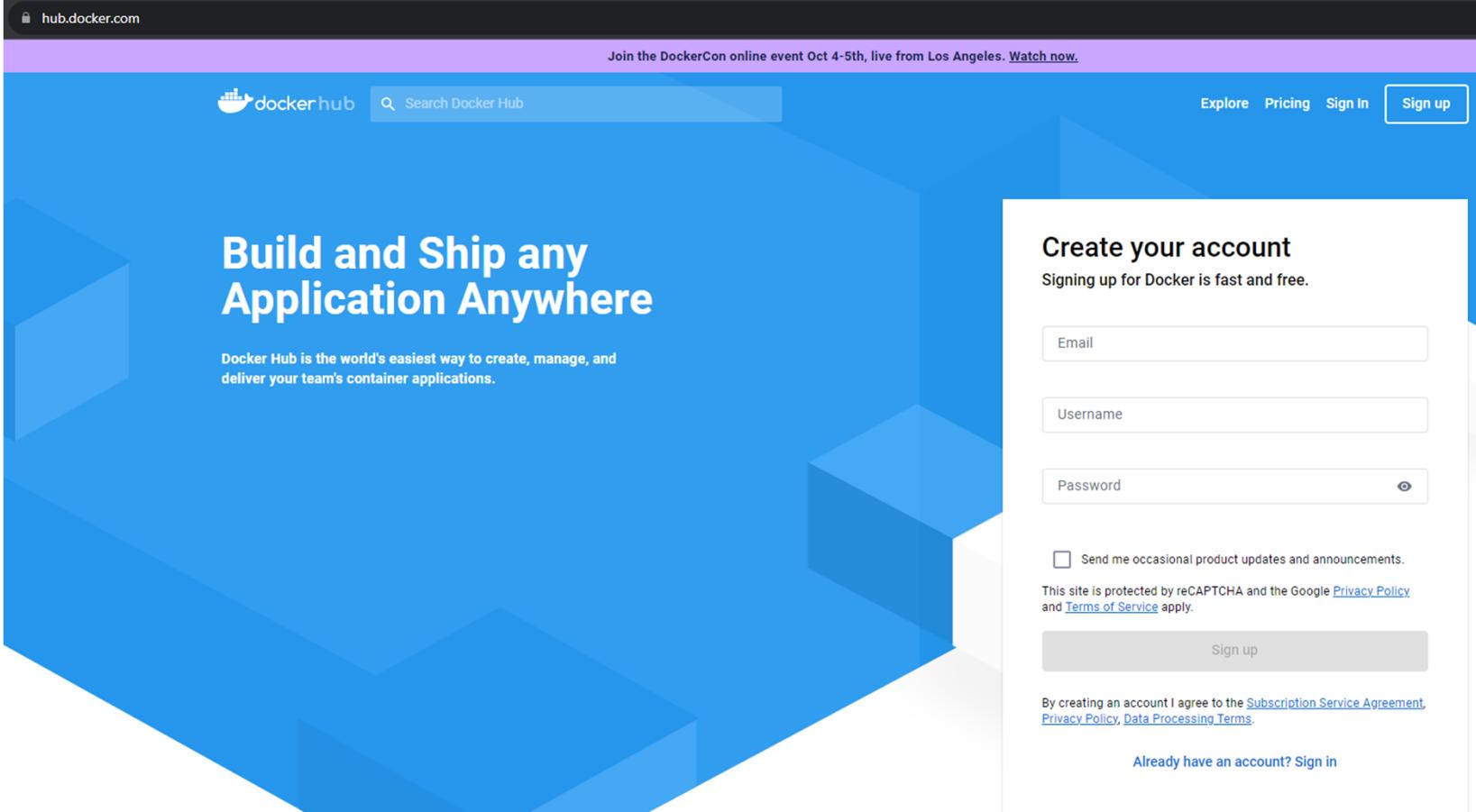


Verificación de que todo esta bien

```
Terminal  
[kralos]  
[~]  
□ docker -v  
Docker version 27.2.0, build 3ab4256  
[kralos]  
[~]  
□ █
```



hub.Docker.com

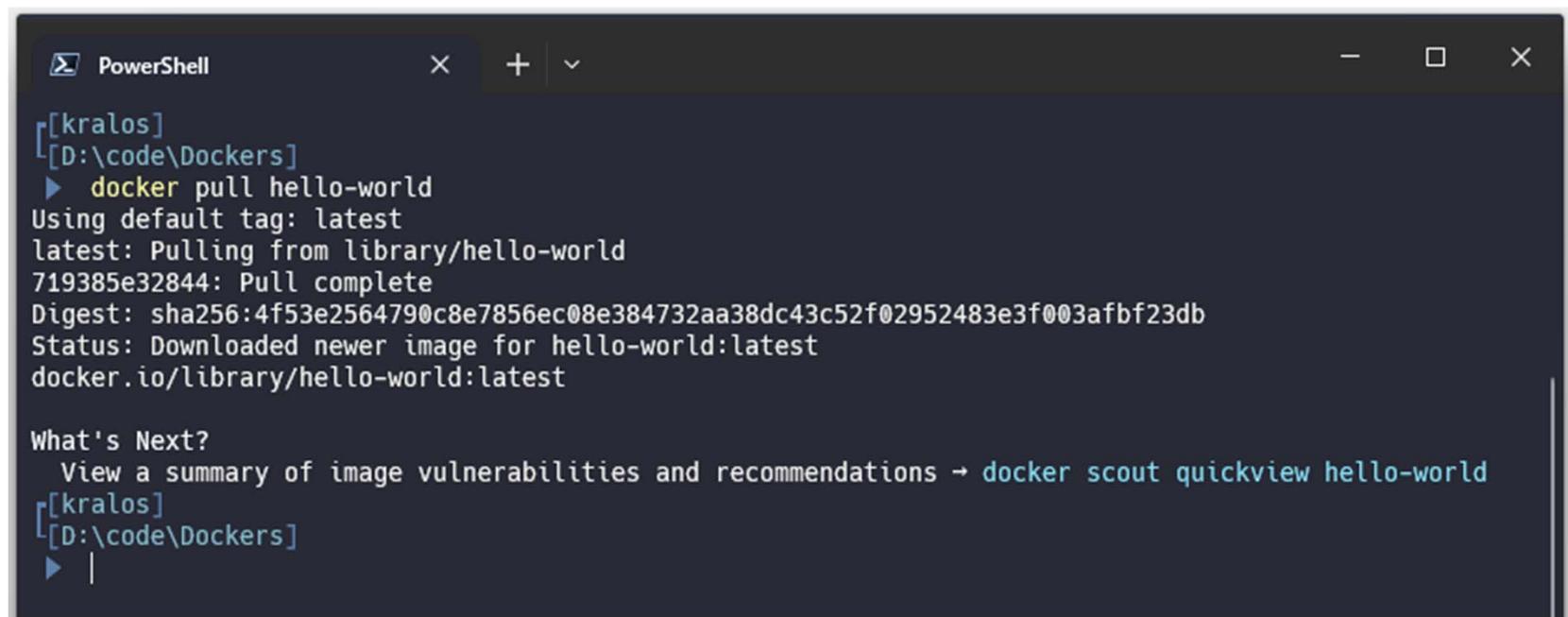


The screenshot shows the Docker Hub sign-up page at hub.docker.com. At the top, there's a purple banner with the text "Join the DockerCon online event Oct 4-5th, live from Los Angeles. [Watch now.](#)". Below the banner, the Docker Hub logo is on the left, followed by a search bar with the placeholder "Search Docker Hub". To the right are links for "Explore", "Pricing", "Sign In", and a prominent "Sign up" button. The main area features a large blue hexagonal graphic on the left with the text "Build and Ship any Application Anywhere" and a description below it: "Docker Hub is the world's easiest way to create, manage, and deliver your team's container applications." On the right, a white form box is titled "Create your account" with the sub-instruction "Signing up for Docker is fast and free." It contains fields for "Email", "Username", and "Password" (with a visibility icon). There's also a checkbox for "Send me occasional product updates and announcements." Below the form, a note states: "This site is protected by reCAPTCHA and the Google [Privacy Policy](#) and [Terms of Service](#) apply." At the bottom of the form is a "Sign up" button. A small note at the very bottom right says "By creating an account I agree to the [Subscription Service Agreement](#), [Privacy Policy](#), [Data Processing Terms](#)." At the very bottom center is a link "Already have an account? Sign in".

El hola mundo de docker

The screenshot shows the Docker Hub interface for the 'hello-world' image. At the top, there's a search bar with 'hello-world' and navigation links for 'Explore', 'Official Images', and 'hello-world'. On the right, there are buttons for 'Explore', 'Pricing', 'Sign In', and 'Sign up'. Below the header, the image card for 'hello-world' is displayed, showing its status as a 'Docker Official Image' with 1B+ pulls and 2.1K stars. It's described as 'Hello World! (an example of minimal Dockerization)'. A 'docker pull hello-world' command is shown in a terminal-like box, along with a copy icon. The page is divided into sections: 'Quick reference' (with maintainer info and help links), 'Supported tags and respective Dockerfile links' (with a note about shared vs simple tags), 'Simple Tags' (listing 'linux', 'nanoserver-ltsc2022', and 'nanoserver-1809'), 'Recent Tags' (listing 'nanoserver-ltsc2022', 'nanoserver-1809', 'nanoserver', 'latest', 'linux', 'nanoserver-1803', 'nanoserver-1709', 'nanoserver-sac2016', and 'nanoserver1709'), and 'About Official Images' (explaining what official images are and why they're important).

Ahora si, primeros pasos

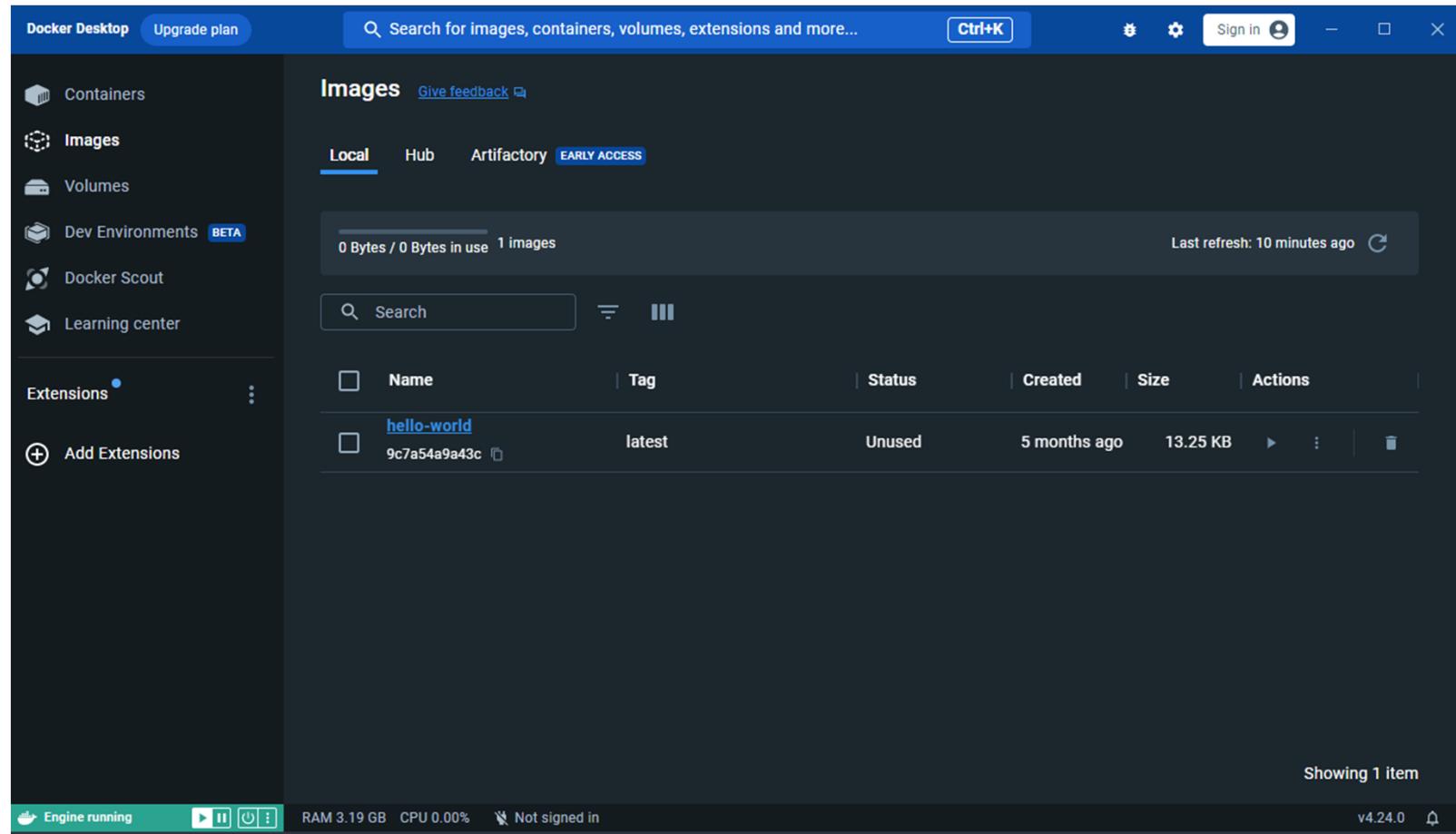


The screenshot shows a PowerShell window with the title bar "PowerShell". The command entered was "docker pull hello-world". The output indicates that the latest tag is being pulled from the library/hello-world repository, resulting in a digest hash and a status message about a newer image being downloaded. A "What's Next?" section at the bottom suggests viewing image vulnerabilities with "docker scout quickview hello-world".

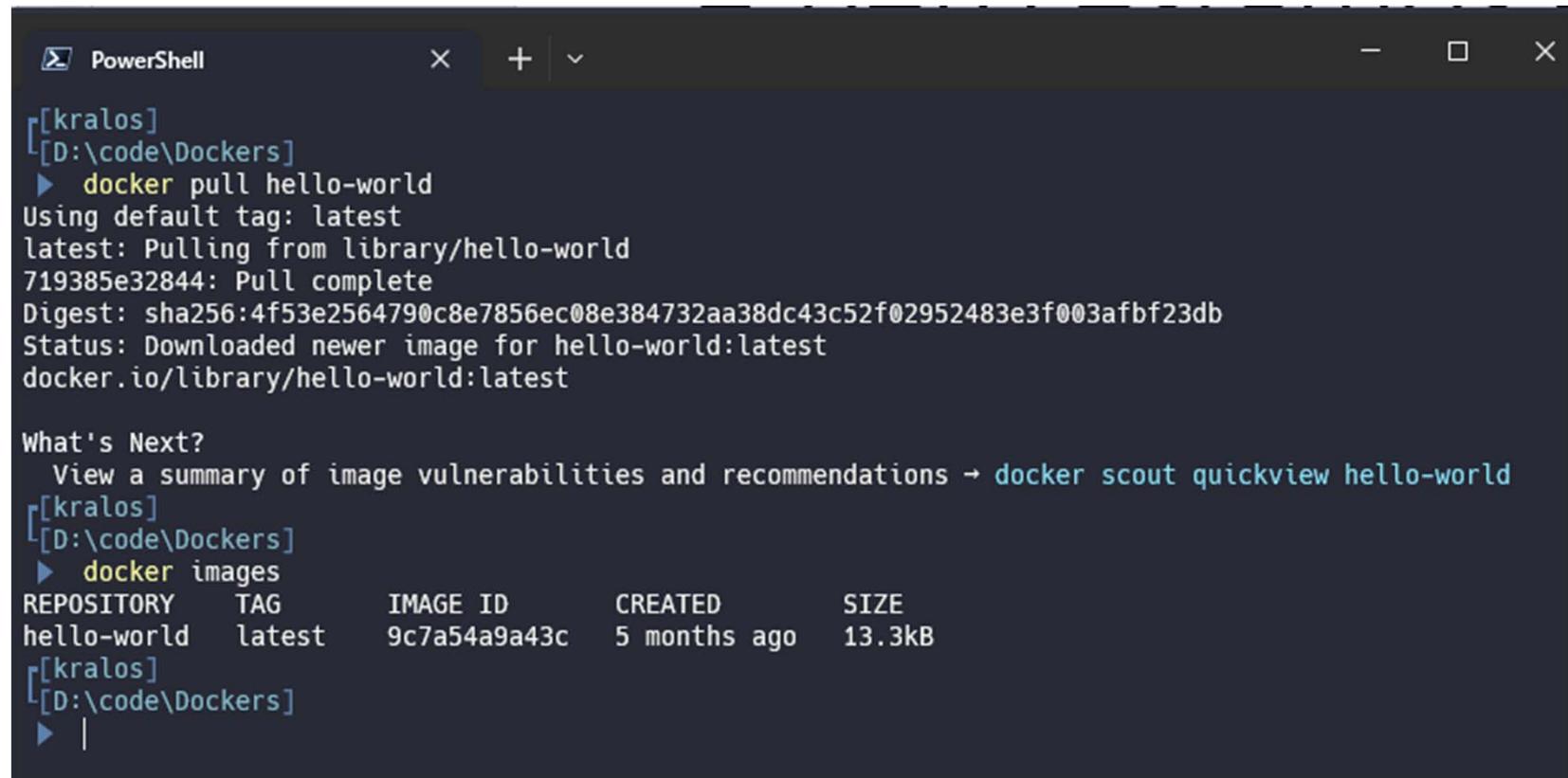
```
[[kralos]
[D:\code\Dockers]
▶ docker pull hello-world
Using default tag: latest
latest: Pulling from library/hello-world
719385e32844: Pull complete
Digest: sha256:4f53e2564790c8e7856ec08e384732aa38dc43c52f02952483e3f003afbf23db
Status: Downloaded newer image for hello-world:latest
docker.io/library/hello-world:latest

What's Next?
  View a summary of image vulnerabilities and recommendations → docker scout quickview hello-world
[[kralos]
[D:\code\Dockers]
▶ |
```

Por si las flies ...



A pero estamos en CLI ...



The screenshot shows a PowerShell window with the following content:

```
[kralos]
[D:\code\Dockers]
▶ docker pull hello-world
Using default tag: latest
latest: Pulling from library/hello-world
719385e32844: Pull complete
Digest: sha256:4f53e2564790c8e7856ec08e384732aa38dc43c52f02952483e3f003afbf23db
Status: Downloaded newer image for hello-world:latest
docker.io/library/hello-world:latest

What's Next?
View a summary of image vulnerabilities and recommendations → docker scout quickview hello-world
[kralos]
[D:\code\Dockers]
▶ docker images
REPOSITORY      TAG          IMAGE ID      CREATED        SIZE
hello-world     latest       9c7a54a9a43c   5 months ago   13.3kB
[kralos]
[D:\code\Dockers]
▶ |
```

Características

- **Autogestión** de los contenedores.
- **Fiabilidad** .
- Aplicaciones **libres** de las **dependencias** instaladas en el sistema anfitrión.
- Capacidad para **desplegar multitud de contenedores** en un mismo equipo físico.
- Puesta en marcha de los servicios en un abrir y cerrar de ojos.
- Contenedores muy **livianos** que facilitan su almacenaje, transporte y despliegue.
- Capacidad para ejecutar una **amplia gama** de **aplicaciones** (prácticamente cualquier cosa que se nos ocurra podrá ejecutarse en un contenedor Docker).

Características

- Compatibilidad **Multi-Sistema** , podremos desplegar nuestros contenedores en multitud de plataformas.
- La aplicación base de Docker **gestionará los recursos existentes** para asignarlos responsablemente entre los contenedores desplegados.
- Establecer una **base** desde la que comenzar nuestros proyectos, lo que nos ahorrará el tiempo de preparar el entorno para cada uno de ellos.
- **Compartir** nuestros contenedores para aumentar los repositorios de Docker así como beneficiarnos de los que comparten los demás.

Componentes y conceptos

- En un principio contamos con una imagen base, sobre la que realizaremos los diferentes cambios. Tras confirmar estos cambios mediante la aplicación Docker, crearemos la imagen que usaremos.
- Esta imagen contiene únicamente las diferencias que hemos añadido con respecto a la base. Cada vez que queramos ejecutar esta imagen necesitaremos la base y las 'capas' de la imagen.

Componentes y conceptos

- Docker se encargará de acoplar la base, la imagen y las diferentes capas con los cambios para darnos el entorno que queremos desplegar para empezar a trabajar.
- Desde su sitio oficial podemos ver que está disponible para Ubuntu, ArchLinux, Gentoo, Fedora, OpenSUSE y FrugalWare , así como desde el propio código binario de la aplicación.

Componentes y conceptos

- La imagen Docker podríamos entenderla como un SO con aplicaciones instaladas (Por ejemplo un OpenSUSE con un paquete ofimático). Sobre esta base podremos empezar a añadir aplicaciones que vayamos a necesitar en otro equipo donde tengamos intención de usar la imagen. Además Docker nos ofrece una forma muy sencilla de actualizar las imágenes que tengamos creadas, así como un sencillo método para crear nuevas imágenes.

Componentes y conceptos

- Contenedor: A diferencia de una máquina virtual que proporciona virtualización de hardware, un contenedor proporciona virtualización ligera a nivel de sistema operativo mediante la abstracción del "espacio del usuario".
- Los contenedores comparten el núcleo del sistema host con otros contenedores.

Componentes y conceptos

- Un contenedor, que se ejecuta en el sistema operativo host, es una unidad de software estándar que empaqueta código y todas sus dependencias, para que las aplicaciones se puedan ejecutar de forma rápida y fiable de un entorno a otro.
- Los contenedores no son persistentes y se activan desde imágenes.

Arquitectura de Docker

- Docker tiene una arquitectura de cliente/servidor.
- Servidor Docker: Es una máquina física o virtual que ejecuta Linux (u otro SO compatible con Docker-Engine).
- Docker Engine: Es una aplicación de cliente/servidor que consta del servicio Docker, una API de Docker que interactúa con el servicio y una interfaz de línea de comandos (CLI) que se comunica con el servicio.

Arquitectura de Docker

- Servicio Docker: Es el que crea y administra imágenes de Docker, mediante el uso de los comandos del cliente. Sirve como el centro de control para la implementación de Docker.
- Cliente Docker: Proporciona la CLI que accede a la API de Docker (una API REST) para comunicar con el servicio de Docker a través de sockets Unix o una interfaz de red.

Arquitectura de Docker

- Objetos de Docker: Son componentes de un despliegue de Docker que ayudan a empaquetar y distribuir aplicaciones. Incluyen imágenes, contenedores, redes, volúmenes, plug-ins y mucho más.
- Contenedores Docker: Son las instancias en tiempo real y en ejecución de las imágenes de Docker. Si bien las imágenes Docker son archivos de solo lectura, los contenedores son contenido ejecutable, efímero y en vivo.

Arquitectura de Docker

- **Imágenes de Docker:** Contienen código fuente de la aplicación ejecutable y todas las herramientas, bibliotecas y dependencias que el código de la aplicación necesita para ejecutar como contenedor. Cuando un desarrollador ejecuta la imagen de Docker, se convierte en una instancia (o varias instancias) del contenedor.

Arquitectura de Docker

- Las imágenes de Docker se componen de capas, y cada capa corresponde a una versión de la imagen. Cada vez que un desarrollador realiza cambios en la imagen, se crea una nueva capa superior y esta capa superior reemplaza a la capa superior anterior como la versión actual de la imagen. Las capas anteriores se guardan para retrocesos o para reutilizarlas en otros proyectos.

Arquitectura de Docker

- Dockerfile: Cada contenedor acoplador comienza con un archivo de texto simple que contiene instrucciones para crear la imagen del contenedor Docker. Dockerfile automatiza el proceso de creación de imágenes Docker. Es esencialmente una lista de instrucciones de la CLI que Docker Engine ejecutará para ensamblar la imagen. La lista de comandos Docker es amplia pero estandarizada: las operaciones Docker funcionan igual independientemente de los contenidos, infraestructura u otras variables del entorno.

Arquitectura de Docker

- Docker Hub: Es el repositorio público de imágenes de Docker, y se autodenomina la biblioteca y comunidad de imágenes de contenedores. Contiene más de 100 000 imágenes de contenedores procedentes de proveedores de software comercial, proyectos de código abierto y desarrolladores individuales. Docker Hub incluye imágenes producidas por Docker, Inc., imágenes certificadas que pertenecen al Docker Trusted Registry y miles de otras imágenes.

Arquitectura de Docker

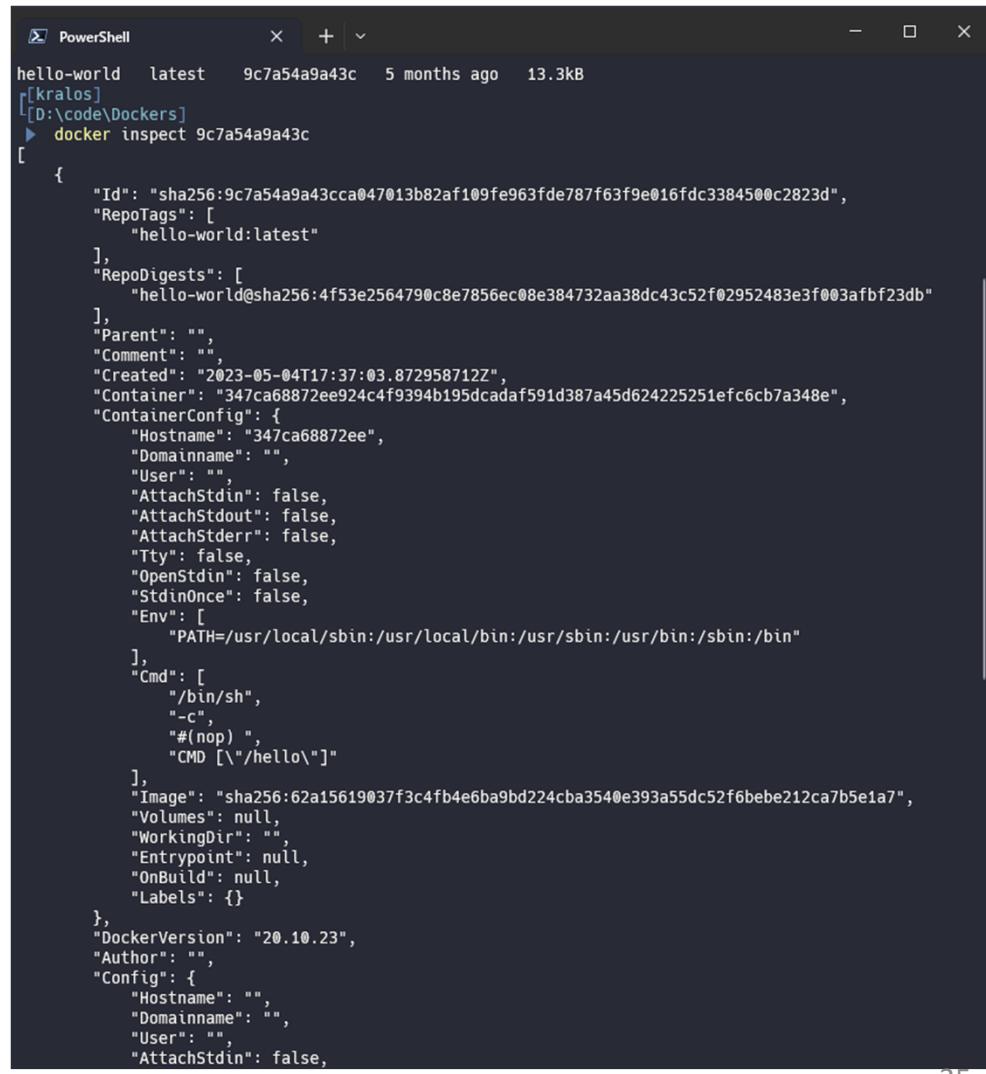
- Docker Desktop: Docker Desktop es una aplicación para Mac o Windows que incluye Docker Engine, Docker CLI Client, Docker Compose, Kubernetes y otros. También proporciona acceso a Docker Hub.
- Registro Docker: es un sistema de almacenamiento y distribución escalable y de código abierto para imágenes Docker. Permite a los desarrolladores rastrear las versiones de imágenes en repositorios mediante el uso de etiquetas para su identificación. Este seguimiento e identificación se logra mediante Git, una herramienta de control de versiones.

Arquitectura de Docker

- Docker Compose: Para gestionar aplicaciones multicontenedor, donde los contenedores se ejecutan en el mismo host de Docker. Docker Compose crea un archivo YAML (.YML) que especifica los servicios que se incluyen en la aplicación, y puede desplegar y ejecutar contenedores con un solo comando. Debido a que la sintaxis YAML es independiente del lenguaje, los archivos YAML se pueden usar en programas escritos en Java, Python, Ruby y muchos otros lenguajes.

Comandos para imágenes docker

Imagenes
docker images
Lista las imágenes almacenadas en el host
docker search imagen
Busca una imagen en el Docker hub
docker inspect id/nombre
Muestra los detalles de una imagen a partir de un id o un nombre
docker pull nombre
Descarga la imagen correspondiente a ese nombre
docker rmi id/nombre
Elimina la imagen correspondiente a ese id o nombre



```
PowerShell
hello-world  latest  9c7a54a9a43c  5 months ago  13.3kB
[kralos]
[D:\code\Dockers]
▶ docker inspect 9c7a54a9a43c
[
  {
    "Id": "sha256:9c7a54a9a43cca047013b82af109fe963fde787f63f9e016fdc3384500c2823d",
    "RepoTags": [
      "hello-world:latest"
    ],
    "RepoDigests": [
      "hello-world@sha256:4f53e2564790c8e7856ec08e384732aa38dc43c52f02952483e3f003afbfb23db"
    ],
    "Parent": "",
    "Comment": "",
    "Created": "2023-05-04T17:37:03.872958712Z",
    "Container": "347ca68872ee924c4f9394b195dcadaf591d387a45d624225251efc6cb7a348e",
    "ContainerConfig": {
      "Hostname": "347ca68872ee",
      "Domainname": "",
      "User": "",
      "AttachStdin": false,
      "AttachStdout": false,
      "AttachStderr": false,
      "Tty": false,
      "OpenStdin": false,
      "StdinOnce": false,
      "Env": [
        "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
      ],
      "Cmd": [
        "/bin/sh",
        "-c",
        "#(nop)",
        "CMD [\"/hello\"]"
      ],
      "Image": "sha256:62a15619037f3c4fb4e6ba9bd224cba3540e393a55dc52f6bebe212ca7b5e1a7",
      "Volumes": null,
      "WorkingDir": "",
      "Entrypoint": null,
      "OnBuild": null,
      "Labels": {}
    },
    "DockerVersion": "20.10.23",
    "Author": "",
    "Config": {
      "Hostname": "",
      "Domainname": "",
      "User": "",
      "AttachStdin": false
    }
  }
]
```

Pull especificando una etiqueta

The screenshot shows the Docker Hub interface for the `hello-world` repository. The `Tags` tab is selected. The page displays two main sections: `latest` and `linux`.

latest Section:

DIGEST	OS/ARCH	VULNERABILITIES	COMPRESSED SIZE
004d23c66201	linux/386	None found	2.65 KB
efd257c8ea08	windows/amd64	None found	114.99 MB
75043f8f1db5	windows/amd64	None found	99.65 MB

A tooltip for the first row (`004d23c66201`) contains the command `docker pull hello-world:latest`.

linux Section:

DIGEST	OS/ARCH	VULNERABILITIES	COMPRESSED SIZE
004d23c66201	linux/386	None found	2.65 KB
7e9b6e7ba284	linux/amd64	None found	2.4 KB
084c3bdd1271	linux/arm/v5	None found	3.57 KB

A tooltip for the first row (`004d23c66201`) contains the command `docker pull hello-world:linux`.

Otro hola mundo

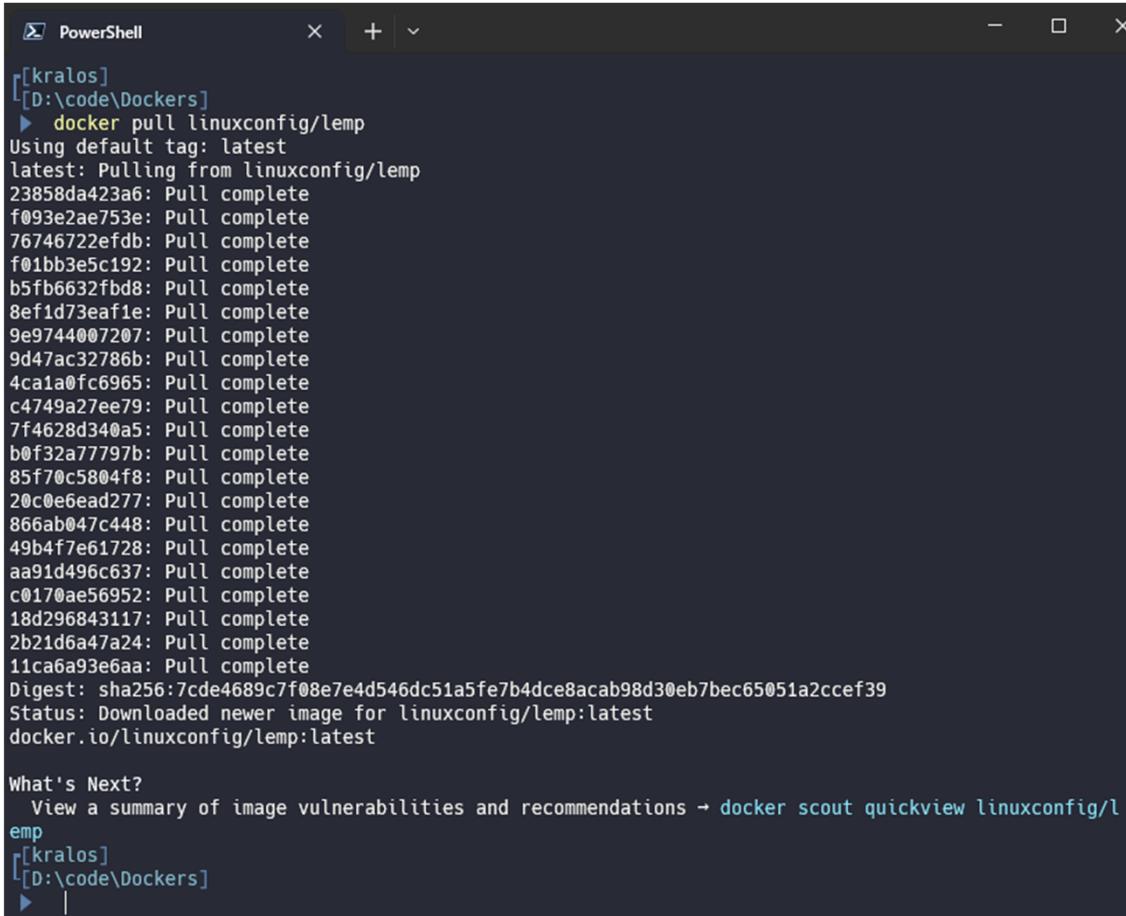
```
[kralos] [D:\code\Dockers]
▶ docker pull hello-world:nanoserver-ltsc2022
nanoserver-ltsc2022: Pulling from library/hello-world
no matching manifest for linux/amd64 in the manifest list entries
[kralos][xERROR]
[D:\code\Dockers]
▶ docker images
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
hello-world     latest   9c7a54a9a43c  5 months ago  13.3kB
linuxconfig/lEMP latest   670ff9140a94  12 months ago 1.05GB
[kralos]
[D:\code\Dockers]
▶ docker pull hello-world:linux
linux: Pulling from library/hello-world
Digest: sha256:726023f73a8fc5103fa6776d48090539042cb822531c6b751b1f6dd18cb5705d
Status: Downloaded newer image for hello-world:linux
docker.io/library/hello-world:linux

What's Next?
 1. Sign in to your Docker account → docker login
 2. View a summary of image vulnerabilities and recommendations → docker scout quickview hello-wor
d:linux
[kralos]
[D:\code\Dockers]
▶ docker images
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
hello-world     latest   9c7a54a9a43c  5 months ago  13.3kB
hello-world     linux    9c7a54a9a43c  5 months ago  13.3kB
linuxconfig/lEMP latest   670ff9140a94  12 months ago 1.05GB
[kralos]
[D:\code\Dockers]
▶ |
```

A descargar imágenes

The screenshot shows the Docker Hub interface for the repository `linuxconfig/lemp`. The top navigation bar includes the Docker Hub logo, a search bar with the query `lemp`, and links for Explore, Pricing, Sign In, and Sign up. Below the header, the repository path `Explore / linuxconfig/lemp` is shown. The main card displays the repository name `linuxconfig/lemp` with a star icon, the owner `linuxconfig`, and a build status indicating it was updated a year ago. It also mentions a stable build LEMP stack environment for fast application deployments and provides a link to the Docker image. The `Image` tab is selected. Below the card, there are two tabs: `Overview` (which is active) and `Tags`. The `Overview` section contains a summary of the LEMP Stack, including a link to the source at <https://linuxconfig.org>. It describes the image as a stable automated build for dynamic PHP applications, mentioning Debian Linux 11, Nginx 1.18.0, MariaDB 10.5.15-MariaDB, and PHP 7.4.30. The `Docker Pull Command` is provided as `docker pull linuxconfig/lemp` with a copy icon. To the right, a `Source Repository` section links to Bitbucket with the repository name `linuxconfig/lemp`.

Se tardo creo ... no vi :S



```
[kralos] [D:\code\Dockers] ▶ docker pull linuxconfig/lEMP
Using default tag: latest
latest: Pulling from linuxconfig/lEMP
23858da423a6: Pull complete
f093e2ae753e: Pull complete
76746722efdb: Pull complete
f01bb3e5c192: Pull complete
b5fb6632fdb8: Pull complete
8ef1d73eaf1e: Pull complete
9e9744007207: Pull complete
9d47ac32786b: Pull complete
4ca1a0fc6965: Pull complete
c4749a27ee79: Pull complete
7f4628d340a5: Pull complete
b0f32a77797b: Pull complete
85f70c5804f8: Pull complete
20c0e6ead277: Pull complete
866ab047c448: Pull complete
49b4f7e61728: Pull complete
aa91d496c637: Pull complete
c0170ae56952: Pull complete
18d296843117: Pull complete
2b21d6a47a24: Pull complete
11ca6a93e6aa: Pull complete
Digest: sha256:7cde4689c7f08e7e4d546dc51a5fe7b4dce8acob98d30eb7bec65051a2cce39
Status: Downloaded newer image for linuxconfig/lEMP:latest
docker.io/linuxconfig/lEMP:latest

What's Next?
  View a summary of image vulnerabilities and recommendations → docker scout quickview linuxconfig/lEMP
[kralos] [D:\code\Dockers] ▶ |
```

Comandos para contenedores

Contenedores

docker create -it --name nombre imagen

Crea un contenedor denominado name a partir de imagen

docker start nombre

Arranca el contenedor denominado nombre

docker stop nombre

Para el contenedor denominado nombre

docker restart nombre

Rearranca el contenedor denominado nombre

docker rm nombre

Elimina el contenedor denominado nombre

docker ps -a

Lista todos los contenedores en ejecución (-a incluye los parados)

docker run -it imagen comando

Arranca y ejecuta el comando en un contenedor de esa imagen en modo interactivo

docker exec -it nombre comando

Ejecuta el comando en el contenedor nombre

docker run -d

Arranca el contenedor en modo daemon

docker run -P

Arranca el contenedor y expone los puertos del contenedor en puertos aleatorios del host

docker run -p

puerto_host:puerto_contenedor

Arranca el contenedor y expone el puerto_contenedor en el puerto_host

docker run -v

directorio_host:directorio_contenedor

Asigna el directorio_host para mapearlo en el directorio_contenedor

docker run --hostname nombre_host

Arranca el contenedor y asigna nombre_host al contenedor

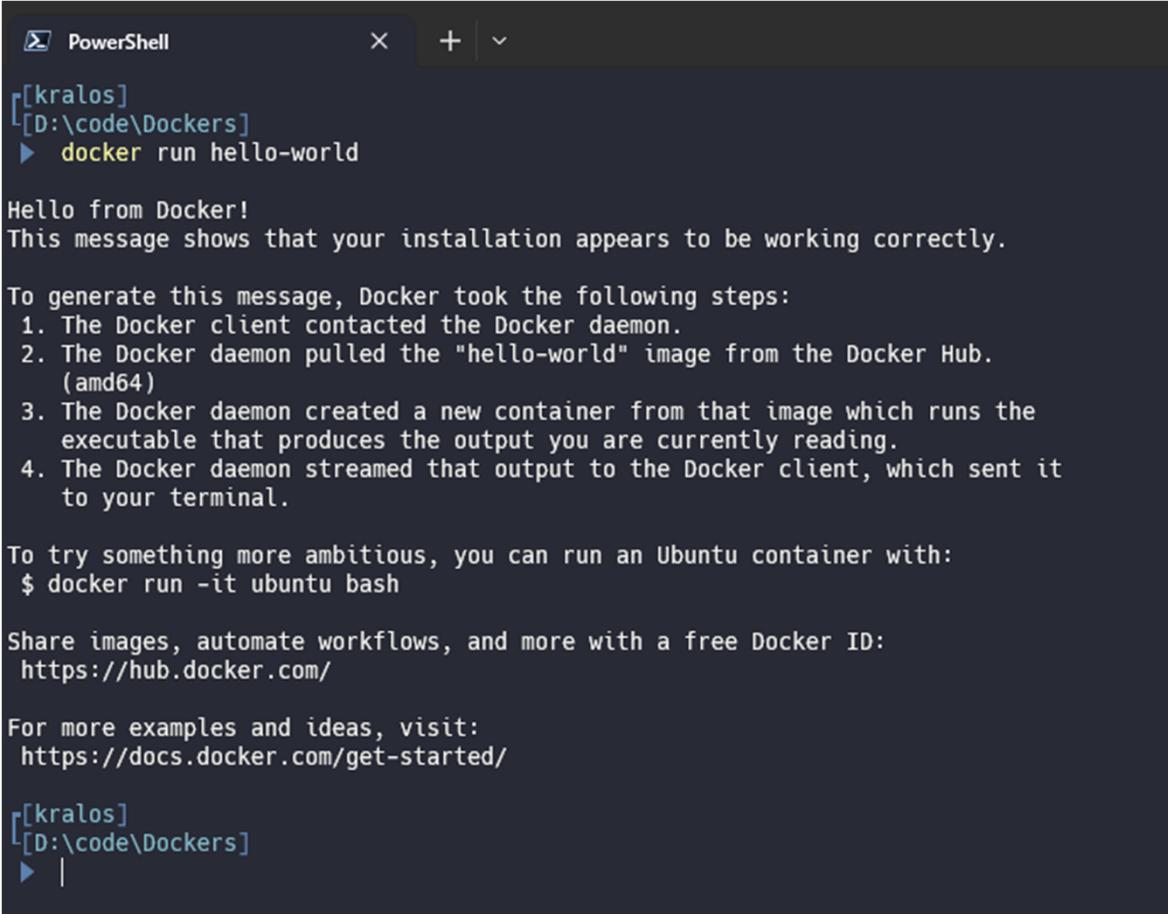
docker run --net=mired

Arranca el contenedor y lo asigna en la red denominada mired

“Crear” un contenedor

```
Σ PowerShell × + ▾ — □ ×
[[kralos]
[D:\code\Dockers]
▶ docker images
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
hello-world     latest   9c7a54a9a43c  5 months ago  13.3kB
hello-world     linux    9c7a54a9a43c  5 months ago  13.3kB
linuxconfig/lEMP  latest   670ff9140a94  12 months ago  1.05GB
[[kralos]
[D:\code\Dockers]
▶ docker create hello-world
2bc45d52863bfde33bc68c0caf3a33b479de42e3bc2d9ddd9575bb57afc9f812
[[kralos]
[D:\code\Dockers]
▶ docker ps -a
CONTAINER ID      IMAGE      COMMAND      CREATED      STATUS      PORTS      NAMES
2bc45d52863b      hello-world      "/hello"    36 seconds ago      Created      gifted_matsumoto
[[kralos]
[D:\code\Dockers]
▶ |
```

Ejecutando un contenedor ..



```
PowerShell          X + | ▾
[kralos]
[D:\code\Dockers]
▶ docker run hello-world

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

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

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

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

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

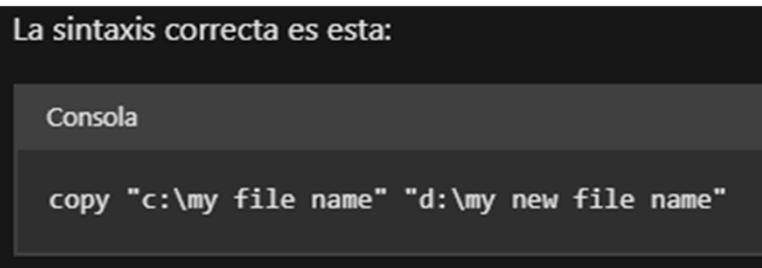
[kralos]
[D:\code\Dockers]
▶ |
```

Cuidado con los espacios

- Use comillas al especificar nombres de archivo largos o rutas de acceso con espacios. Por ejemplo, escribir el comando

```
copy c:\my file name d:\my new file name
```

- El símbolo del sistema da como resultado el siguiente mensaje de error: El sistema no puede encontrar el archivo especificado.



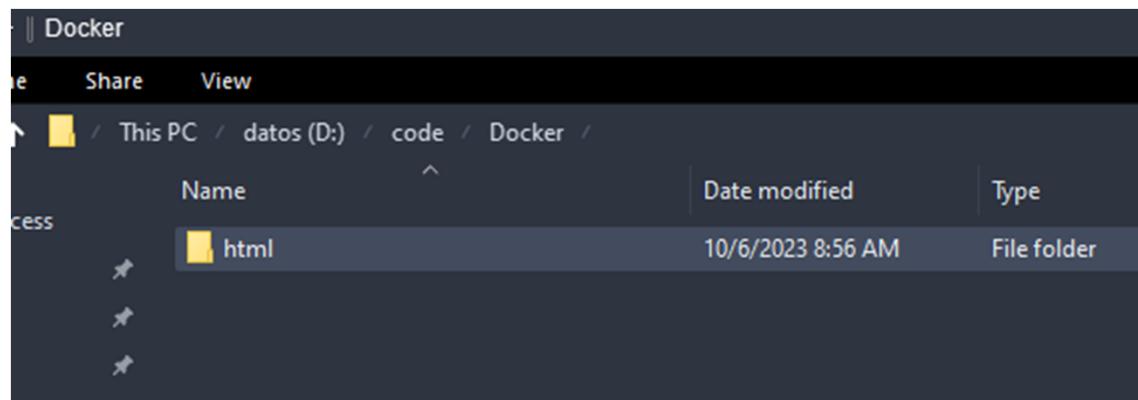
The image shows a terminal window with a dark background and light-colored text. At the top, it says "La sintaxis correcta es esta:". Below that, it says "Consola". At the bottom, it shows a command line input: "copy "c:\my file name" "d:\my new file name"".

```
La sintaxis correcta es esta:  
Consola  
copy "c:\my file name" "d:\my new file name"
```

Ejemplo de contenedor LEMP

1. Crear un directorio llamado “html” en SO anfitrión.

D:\code\Docker\html

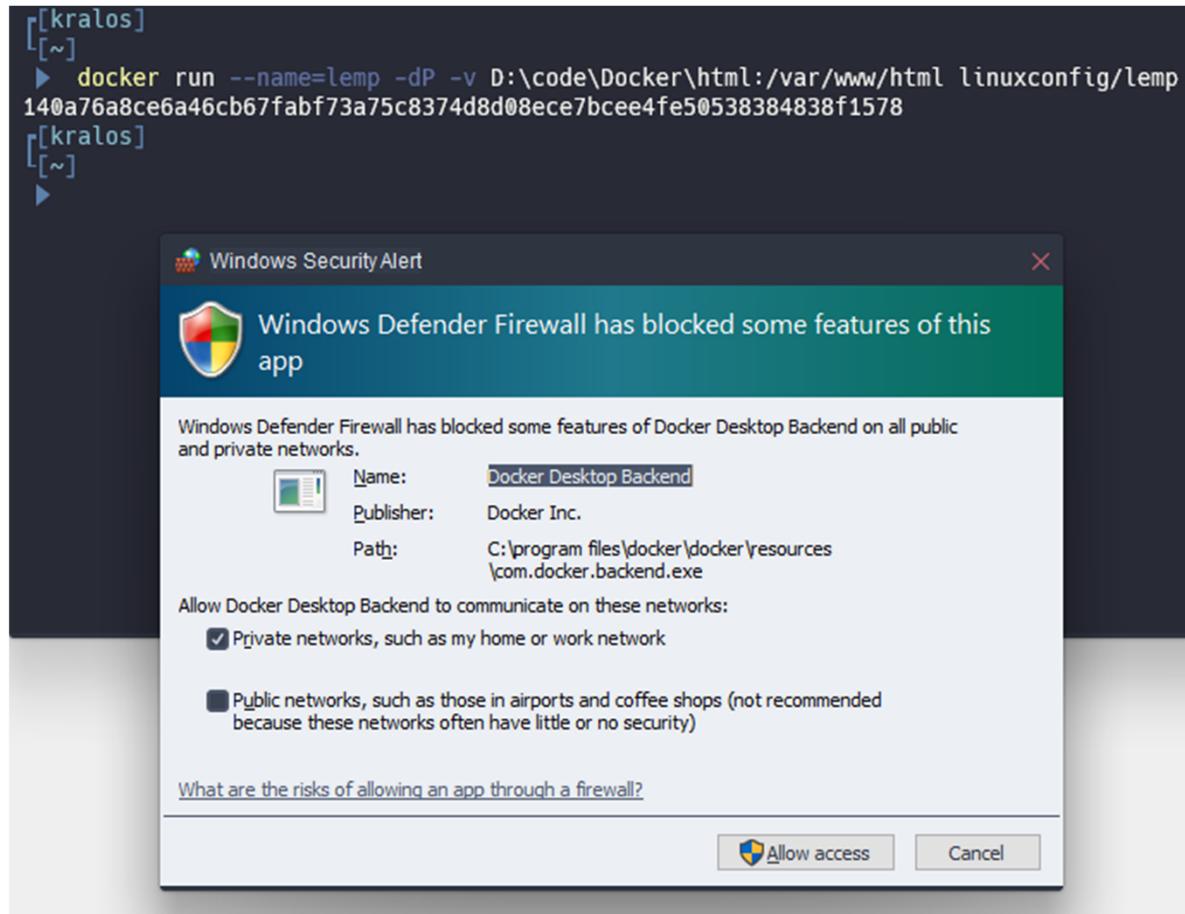


Ejemplo de contenedor LEMP

2. Crear y ejecutar un contendor, además se especifica el directorio referenciado entre la PC local y el contendor a crearse.

```
docker run --name=lemp -dP -v  
D:\code\Docker\html:/var/www/html linuxconfig/lemp
```

Ejemplo de contenedor LEMP



Ejemplo de contenedor LEMP

The screenshot shows the Docker Desktop interface. On the left, there's a sidebar with options like Containers, Images, Volumes, Dev Environments (BETA), Docker Scout, Learning center, Extensions, and Add Extensions. The main area is titled "Containers" and displays the following information:

- Container CPU usage: 0.03% / 400% (4 cores allocated)
- Container memory usage: 120.1MB / 3.61GB
- Search bar and filter: Only show running containers
- A table listing three stopped containers and one running container:

Name	Image	Status	CPU (%)	Port(s)	Last started	Actions
nostalgic_moser 2f5eba4f771b	hello-world	Exited	0%		2 days ago	[More]
gifted_matsumoto 2bc45d52863b	hello-world	Exited	0%		2 days ago	[More]
lemp 140a76a8ce6a	linuxconfig/lemp	Running	0.03%	32768:3306	34 minutes ago	Show all ports (2) [More]
- A message at the bottom says "Showing 3 items"
- A "Walkthroughs" section with two cards:
 - "What is a container?" (5 mins) with a question mark icon.
 - "How do I run a container?" (6 mins) with a terminal icon showing commands: FROM node, RUN mkdir -p /app, WORKDIR /app, COPY packa[REDACTED]
- At the bottom, status icons include "Engine running", "RAM 3.03 GB", "CPU 0.25%", "Not signed in", version "v4.24.0", and a notification bell with "1".

Ejemplo de contenedor LEMP

3. Entramos al servidor web, pero antes necesitamos saber en que Puerto esta mapeado.

Vemos que hay dos puertos abiertos:

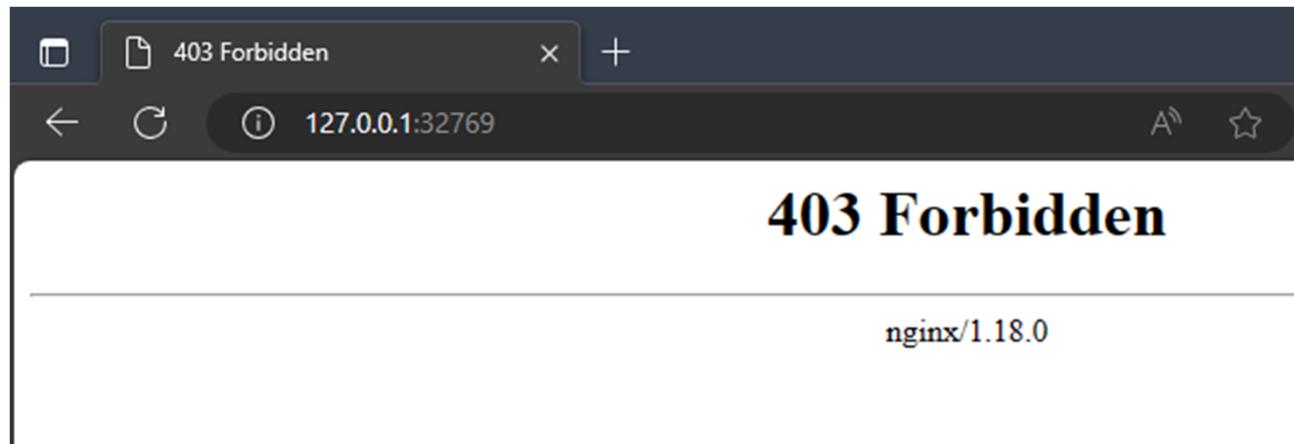
32269 para el servidor web

32768 para el servidor de base de dato

```
▶ docker port lemp
80/tcp -> 0.0.0.0:32769
3306/tcp -> 0.0.0.0:32768
```

Ejemplo de contenedor LEMP

- El servidor web esta activo pero no tiene nada ...



Ejemplo de contenedor LEMP

- El código se debe colocar el directorio “html” creado en el paso 1.
- Podemos usar nuestro editor favorito.

The screenshot shows a code editor on the left and a terminal window on the right. The code editor displays the file `index.php` with the following content:

```
Run Terminal Help
index.php X
index.php > html > body > p
1  <!DOCTYPE html>
2  <html lang="es">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Docker</title>
7  </head>
8  <body>
9      <p>Hola contenedor</p>
10 </body>
11 </html>
```

The terminal window is titled "Docker" and shows the IP address "127.0.0.1:32769". It contains the text "Hola contenedor", which is the output of the PHP script.

Ejemplo de contenedor LEMP

4. Ingresamos a la línea de comandos del contenedor

```
▶ docker exec -it lemp /bin/bash
root@140a76a8ce6a:/# ls
bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin
root@140a76a8ce6a:/# pwd
/
root@140a76a8ce6a:/# whoami
root
root@140a76a8ce6a:/# ls /var/www/html/
index.php
root@140a76a8ce6a:/# cat /var/www/html/index.php
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Docker</title>
</head>
<body>
    <p>Hola contenedor</p>
</body>
</html>root@140a76a8ce6a:/# |
```

Actualizaciones

The screenshot shows a dark-themed software interface for Docker Desktop. At the top, the title "Software updates" is displayed in white. Below it, a message states, "You're currently on version 4.24.0 (122432). The latest version is 4.24.2 (124339)." A list of four bullet points details the new features available in the latest version:

- New Notification center available to all users.
- Compose File Watch available to all users. For more information, see [Use Compose Watch](#).
- Resource Saver available to all users. This feature can be configured from the Resources tab in Settings. For more information see [Docker Desktop's Resource Saver mode](#).
- View and manage options in the Docker Engine state, with pause, stop, and resume, directly from the Docker Dashboard.

Un poco de detalle

```
docker run --name=lemp -dP -v D:\code\Docker\html:/var/www/html linuxconfig/lemp
```

--name=nombre_de_mi contenedor

-dP

d hace que el contenedor corra en segundo plano
(daemonizará el contenedor)

P para decirle a Docker que elija un puerto

-v Cuando queremos montar un volúmen en un contenedor
ruta_origen:ruta_destino

“linuxconfig/lemp” es el nombre de la imagen a usar

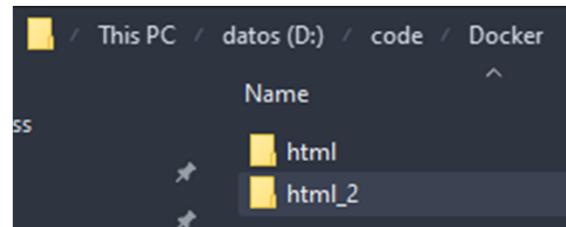
Mapear puertos

`-p puerto_PC_anfitrión: puerto_contendor`

El argumento `-p 1234:1234` le indica a Docker que tiene que hacer un port forwarding del contenedor hacia el puerto 1234 de la máquina host.

Para mas información ejecutar en su terminal el comando siguiente: `docker run --help`

Practica 2



Two browser windows titled 'Docker' are shown. The top window has the URL 'localhost:32769' and displays the text 'Hola contenedor'. The bottom window has the URL 'localhost:8081' and displays the text 'Hola contenedor estoy en el 2'.

```
docker run --name=lemp2 -d -p 8081:80 -v D:\code\ Docker\html_2:/var/www/html linuxconfig/lemp
```

A screenshot of a PowerShell terminal window. The session starts with '[kralos] [~]'. The user runs the command 'docker run --name=lemp2 -d -p 8081:80 -v D:\code\ Docker\html_2:/var/www/html linuxconfig/lemp', which creates a container with ID 'f3c176d990da9f34e4177e0a1556500e4ea3900e4928bdf3897e29b915c6c31c'. The user then runs 'docker ps -a' to list all containers, showing the newly created 'lemp2' container along with others like 'linuxconfig/lemp' and 'hello-world'.

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
f3c176d990da	linuxconfig/lemp	"supervisord"	8 seconds ago	Up 4 seconds	3306/tcp, 0.0.0.0:8081->80/tcp	lemp2
140a76a8ce6a	linuxconfig/lemp	"supervisord"	12 days ago	Up 2 minutes	0.0.0.0:32769->80/tcp, 0.0.0.0:32768->3306/tcp	lemp
2f5eba4f771b	hello-world	"/hello"	13 days ago	Exited (0) 13 days ago		nostalgic_moser
2bc45d52863b	hello-world	"/hello"	13 days ago	Exited (0) 13 days ago		gifted_matsumoto

Comportamiento de los contenedores

docker logs lemp

```
[kralos]
[~]
▶ docker logs lemp
/usr/lib/python3/dist-packages/supervisor/options.py:474: UserWarning: Supervisord is running as root and it is searching for its configuration file in default locations (including its current working directory); you probably want to specify a "-c" argument specifying an absolute path to a configuration file for improved security.
    self.warnings.warn(
2023-10-06 16:45:54,397 CRIT Supervisor is running as root. Privileges were not dropped because no user is specified in the config file. If you intend to run as root, you can set user=root in the config file to avoid this message.
2023-10-06 16:45:54,399 INFO Included extra file "/etc/supervisor/conf.d/supervisor-lemp.conf" during parsing
2023-10-06 16:45:54,421 INFO RPC interface 'supervisor' initialized
2023-10-06 16:45:54,422 CRIT Server 'unix_http_server' running without any HTTP authentication checking
```

```
▶ docker logs lemp2
/usr/lib/python3/dist-packages/supervisor/options.py:474: UserWarning: Supervisord is running as root and it is searching for its configuration file in default locations (including its current working directory); you probably want to specify a "-c" argument specifying an absolute path to a configuration file for improved security.
    self.warnings.warn(
2023-10-18 18:20:06,537 CRIT Supervisor is running as root. Privileges were not dropped because no user is specified in the config file. If you intend to run as root, you can set user=root in the config file to avoid this message.
2023-10-18 18:20:06,537 INFO Included extra file "/etc/supervisor/conf.d/supervisor-lemp.conf" during parsing
2023-10-18 18:20:06,540 INFO RPC interface 'supervisor' initialized
2023-10-18 18:20:06,543 CRIT Server 'unix_http_server' running without any HTTP authentication checking
2023-10-18 18:20:06,543 INFO supervisord started with pid 1
2023-10-18 18:20:07,548 INFO spawned: 'mysqld' with pid 7
2023-10-18 18:20:07,550 INFO spawned: 'nginx' with pid 8
2023-10-18 18:20:07,551 INFO spawned: 'php-fcgi' with pid 9
2023-10-18 18:20:08,693 INFO success: mysqld entered RUNNING state, process has stayed up for > than 1 seconds (startsecs)
2023-10-18 18:20:08,693 INFO success: nginx entered RUNNING state, process has stayed up for > than 1 seconds (startsecs)
2023-10-18 18:20:08,693 INFO success: php-fcgi entered RUNNING state, process has stayed up for > than 1 seconds (startsecs)
[kralos]
[~]
▶ |
```

Variables de entorno

Las variables de entorno se usan para definir valores que están disponibles para los procesos que se ejecutan dentro del contenedor. Estas variables se definen al construir una imagen o al ejecutar un contenedor.

-e variable_entorno:valor_nuestro
es para asignar valores a las variables de entorno. Esas variables están definidas dentro de la imagen, por lo que nosotros le damos valor, para que el contenedor a ejecutar, use esos datos.

Usaremos la imagen de mongo

- docker pull mongo
- docker run --name=mongo_server -d -p 8082:27017 mongo

```
[kralos]
[~]
▶ docker pull mongo
Using default tag: latest
latest: Pulling from library/mongo
43f89b94cd7d: Pull complete
54a7480baa9d: Pull complete
7f9301fbdf7df: Pull complete
5e4470f2e90f: Pull complete
40d046ff8fd3: Pull complete
e062d62b861e: Pull complete
72919e34fde8: Pull complete
ab22810dfc64: Pull complete
fb05c29fbdf5: Pull complete
Digest: sha256:d341a86584b96eb665345a8f5b35fba8695ee1d0618fd012ec4696223a3d6c62
Status: Downloaded newer image for mongo:latest
docker.io/library/mongo:latest
```

Practica 3

The screenshot shows a web browser window with the URL `hub.docker.com/_/mongo` in the address bar. The main content is titled "Environment Variables". It explains that when starting the `mongo` image, environment variables can be passed via the `docker run` command line. It notes that if a data directory already contains a database, existing databases will remain untouched. Two environment variables are highlighted: `MONGO_INITDB_ROOT_USERNAME` and `MONGO_INITDB_ROOT_PASSWORD`.

When you start the `mongo` image, you can adjust the initialization of the MongoDB instance by passing one or more environment variables on the `docker run` command line. Do note that none of the variables below will have any effect if you start the container with a data directory that already contains a database: any pre-existing database will always be left untouched on container startup.

`MONGO_INITDB_ROOT_USERNAME` , `MONGO_INITDB_ROOT_PASSWORD`

```
docker run --name=monguito -d -p 8083:27017  
-e MONGO_INITDB_ROOT_USERNAME=nico  
-e MONGO_INITDB_ROOT_PASSWORD=12345678 mongo
```

Practica 3

```
[kralos]--[¶main = ● ]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\nodejs\proyecto_1]
● ▶ docker ps -a
CONTAINER ID   IMAGE      COMMAND      CREATED      STATUS          PORTS      NAMES
f3c176d990da   linuxconfig/lemp "supervisord"  23 hours ago  Exited (137) 23 hours ago
140a76a8ce6a   linuxconfig/lemp "supervisord"  13 days ago   Exited (137) 23 hours ago
2f5eba4f771b   hello-world    "/hello"     2 weeks ago   Exited (0) 2 weeks ago
2bc45d52863b   hello-world    "/hello"     2 weeks ago   Exited (0) 2 weeks ago
[kralos]--[¶main = ● ]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\nodejs\proyecto_1]
● ▶ docker ps
● CONTAINER ID   IMAGE      COMMAND      CREATED      STATUS      PORTS      NAMES
[kralos]--[¶main = ● ]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\nodejs\proyecto_1]
● ▶ docker run --name=monguito -d -p 8083:27017 -e MONGO_INITDB_ROOT_USERNAME=nico -e MONGO_INITDB_ROOT_PASSWORD=12345678 mongo
29293400974ad156801a24f8cc0da3ff3158116250386c3bc7d692daa3cb9e5a
[kralos]--[¶main = ● ]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\nodejs\proyecto_1]
● ▶ docker ps
CONTAINER ID   IMAGE      COMMAND      CREATED      STATUS      PORTS      NAMES
29293400974a   mongo      "docker-entrypoint.s..."  About a minute ago  Up About a minute  0.0.0.0:8083->27017/tcp  monguito
[kralos]--[¶main = ● ]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\nodejs\proyecto_1]
○ ▶ █
```

Practica 3

```
code > nodejs > proyecto_1 > packagejson > ...
You, 2 seconds ago | 1 author (You)
1 { You, 21 hours ago * other project
2   "name": "mongoapp",
3   "version": "1.0.0",
4   "description": "",
5   "main": "index.js",
6   "type": "module",
7     ▶ Debug
8   "scripts": {
9     "test": "echo \"Error: no test specified\" && exit 1"
10  },
11  "author": "",
12  "license": "ISC",
13  "dependencies": {
14    "express": "4.18.1",
15    "mongoose": "6.12.1"
16  }
17 } <- #1-16 { "name": "mongoapp", "version": "1.0.0", "description": "", ...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS
[kralos]--[¶main = ●]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\nodejs\proyecto_1]
● ► npm install
added 87 packages, and audited 88 packages in 2m
13 packages are looking for funding
  run `npm fund` for details
2 vulnerabilities (1 moderate, 1 critical)
● To address all issues, run:
  npm audit fix --force
● Run `npm audit` for details.
[kralos]--[¶main = ●]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\nodejs\proyecto_1]
● ► npm audit fix --force
● npm WARN using --force. Recommended protections disabled.
npm WARN audit Updating mongoose to 6.12.1, which is outside your stated dependency range.
added 82 packages, removed 2 packages, changed 4 packages, and audited 168 packages in 47s
14 packages are looking for funding
  run `npm fund` for details
found 0 vulnerabilities
[kralos]--[¶main = ●]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\nodejs\proyecto_1]
● ► [ ]
```

code > nodejs > proyecto_1 > packagejson > {} dependencies

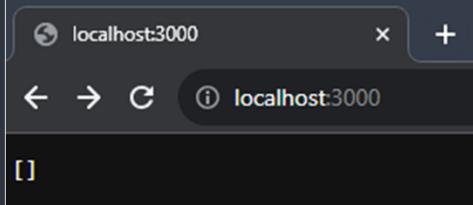
You, 21 hours ago | 1 author (You)

```
1 {
2   "name": "mongoapp",
3   "version": "1.0.0",
4   "description": "",
5   "main": "index.js",
6   "type": "module",
7     ▶ Debug
8   "scripts": {
9     "test": "echo \"Error: no test specified\" && exit 1"
10  },
11  "author": "",
12  "license": "ISC",
13  "dependencies": []
14    "express": "4.18.1",
15    "mongoose": "6.4.1"
16 } <- #1-16 { "name": "mongoapp", "version": "1.0.0", "description": "", ...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS
code > nodejs > proyecto_1 > JS index.js M X
code > nodejs > JS index.js > ...
You, 4 minutes ago | 1 author (You)
1 import express from 'express'
2 import mongoose from 'mongoose'
3
4 const Animal = mongoose.model('Animal', new mongoose.Schema({
5   tipo: String,
6   estado: String,
7 }))
8
9 const app = express()
10
11 mongoose.connect('mongodb://nico:12345678@127.0.0.1:8083/miapp?authSource=admin')
12
13 app.get('/', async (_req, res) => {
14   console.log('listando... chanchitos...')
15   const animales = await Animal.find();
16   return res.send(animales)
17 })
18 app.get('/crear', async (_req, res) => {
19   console.log('creando...')
20   await Animal.create({ tipo: 'Chanchito', estado: 'Feliz' })
21   return res.send('ok')
22 })
23
24 app.listen(3000, () => console.log('listening...'))
```

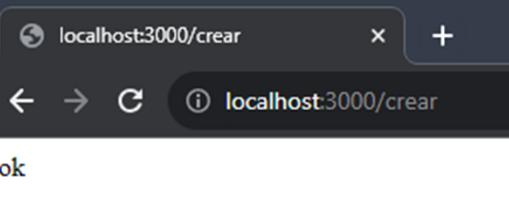
25

Practica 3

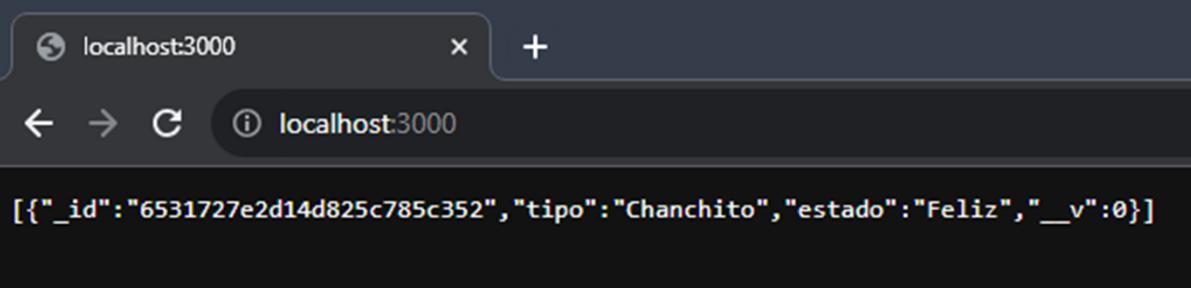
```
[kralos]--[ψmain ≡ ● ]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\nodejs\proyecto_1]
o ▶ node .\index.js
listening...
listando... chanchitos...
█
```



```
[kralos]--[ψmain ≡ ● ]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\nodejs\proyecto_1]
o ▶ node .\index.js
listening...
listando... chanchitos...
creando...
█
```



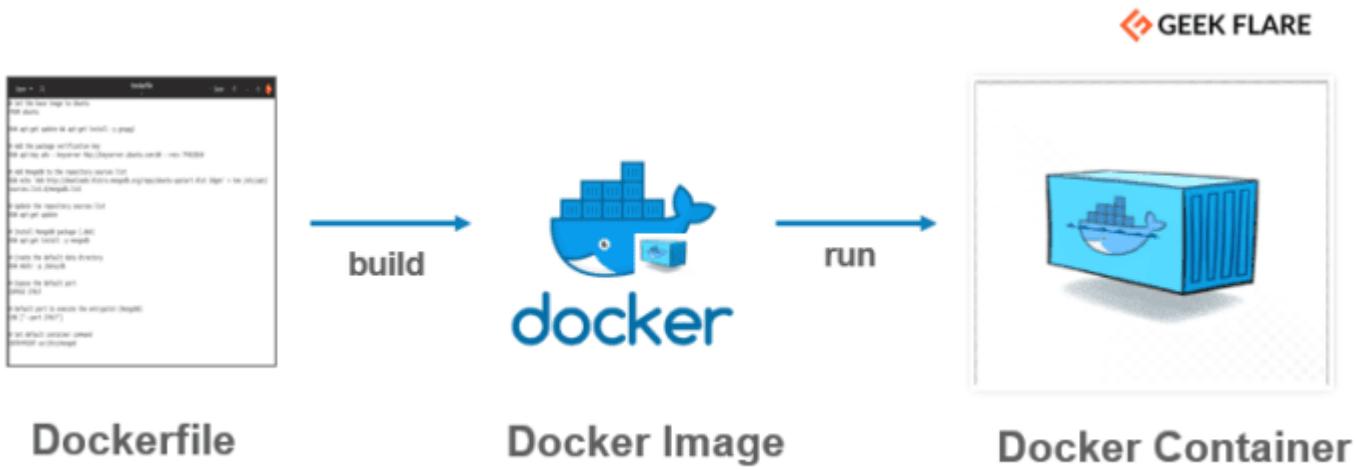
```
[kralos]--[ψmain ≡ ● ]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\nodejs\proyecto_1]
o ▶ node .\index.js
listening...
listando... chanchitos...
creando...
listando... chanchitos...
█
```



Dockerfile

- Es un archivo de texto simple que incluye una serie de instrucciones que se necesitan ejecutar de manera consecutiva para cumplir con los procesos necesarios para la creación de una nueva imagen.
- Este conjunto de instrucciones serán los encargados de indicar los pasos a seguir para el ensamblaje de los elementos necesarios para el desarrollo de un contenedor en Docker.

Dockerfile



Parámetros en Dockerfile

- **FROM:** es la primera instrucción. Establecer la imagen sobre la que los pasos e imágenes siguientes se desarrollan en el sistema.
- **ENV:** Indica las variables de entorno que se necesitan en el proceso de construcción de una imagen en Docker y permite la ejecución de los contenedores y sus labores en el sistema.
- **RUN:** Ejecutar una instrucción incluida en la línea de comandos de la imagen durante su proceso de construcción. El run dockerfile puede escribirse en formato SHELL o bajo la opción de escritura EXEC.

Parámetros en Dockerfile

- **ADD:** Copia de archivos, directorios y una imagen en Dockerfile. Crea una nueva capa de imagen.
- **EXPOSE:** Definición de las asignaciones referentes a los puertos para los contenedores de la plataforma que se encuentren en su etapa de ejecución.
- **CMD:** Función similar al comando RUN, pero se ejecuta sólo después de instanciar el contenedor.
- **ENTRYPOINT:** Se dirige a su aplicación por defecto en la imagen cuando se crea el contenedor.

Parámetros en Dockerfile

Nota: Se puede cambiar ADD por **COPY** ya que ADD es para operaciones de copiado más complejas, y COPY copiamos de manera simple.

Así para poner comentarios

Sin comentarios

Dockerfile en VScode



docker build

```
$ docker build -h

Usage: docker build [OPTIONS] PATH | URL | -

Build an image from a Dockerfile

--build-arg=[]                      Set build-time variables
--cpu-shares=0                        CPU shares (relative weight)
--cgroup-parent=                       Optional parent cgroup for the container
--cpu-period=0                         Limit the CPU CFS (Completely Fair Scheduler) period
--cpu-quota=0                          Limit the CPU CFS (Completely Fair Scheduler) quota
--cpuset-cpus=                         CPUs in which to allow execution (0-3, 0,1)
--cpuset-mems=                         MEMs in which to allow execution (0-3, 0,1)
--disable-content-trust=true          Skip image verification
-f, --file=                            Name of the Dockerfile (Default is 'PATH/Dockerfile')
--force-rm=false                       Always remove intermediate containers
--help=false                           Print usage
-m, --memory=                          Memory limit
--memory-swap=                         Total memory (memory + swap), '-1' to disable swap
--no-cache=false                       Do not use cache when building the image
--pull=false                           Always attempt to pull a newer version of the image
-q, --quiet=false                      Suppress the verbose output generated by the containers
--rm=true                             Remove intermediate containers after a successful build
-t, --tag=                            Repository name (and optionally a tag) for the image
--ulimit=[]                            Ulimit options
```

Dockerfile {ejemplo}

The screenshot shows the Docker Hub page for the **alpine** image. At the top, there's a blue hexagonal icon with a white mountain-like symbol. To its right, the name **alpine** is displayed, followed by a green Docker Official Image badge, a download count of **1B+**, and a star rating of **10K+**. Below this, a subtitle reads: "A minimal Docker image based on Alpine Linux with a complete package index and only 5 MB in size!" A search bar at the top right contains the query **docker pull alpine** with a copy icon to its right.

Overview **Tags**

Quick reference

- Maintained by:
Natanael Copa (an Alpine Linux maintainer)
- Where to get help:
the Docker Community Slack, Server Fault, Unix & Linux, or Stack Overflow

Supported tags and respective Dockerfile links

- [20230901](#) , [edge](#)
- [3.18.4](#) , [3.18](#) , [3](#) , [latest](#)
- [3.17.5](#) , [3.17](#)
- [3.16.7](#) , [3.16](#)
- [3.15.10](#) , [3.15](#)

Recent Tags

[latest](#) [3.18.4](#) [3.18](#) [3](#) [edge](#) [20230901](#) [3.18.3](#)
[3.17.5](#) [3.17](#) [3.16.7](#)

About Official Images

Docker Official Images are a curated set of Docker open source and drop-in solution repositories.

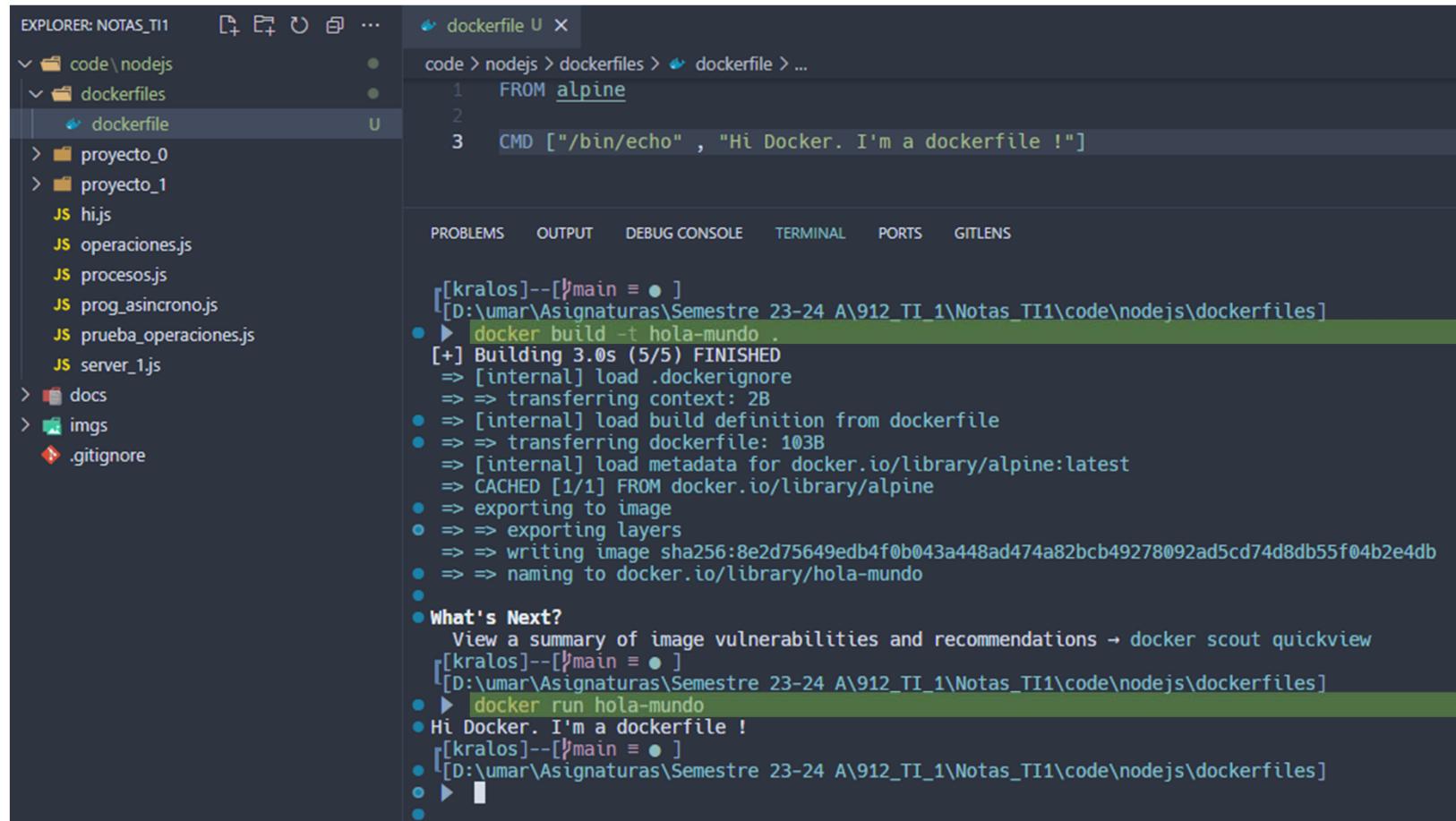
Why Official Images?
These images have clear documentation, promote best practices, and are designed for the most common use cases.

Dockerfile {ejemplo}

```
[kralos]--[✗ main = ● ]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\nodejs\dockerfiles]
▶ docker images
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
mongo           latest   ee3b4d1239f1  7 days ago   748MB
hello-world     latest   9c7a54a9a43c  5 months ago  13.3kB
hello-world     linux    9c7a54a9a43c  5 months ago  13.3kB
linuxconfig/lemp latest   670ff9140a94  13 months ago 1.05GB
[kralos]--[✗ main = ● ]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\nodejs\dockerfiles]
▶ docker pull alpine
Using default tag: latest
latest: Pulling from library/alpine
96526aa774ef: Pull complete
Digest: sha256:eece025e432126ce23f223450a0326fbebe39cdf496a85d8c016293fc851978
Status: Downloaded newer image for alpine:latest
docker.io/library/alpine:latest

What's Next?
View a summary of image vulnerabilities and recommendations → docker scout quickview alpine
[kralos]--[✗ main = ● ]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\nodejs\dockerfiles]
▶ docker images
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
mongo           latest   ee3b4d1239f1  7 days ago   748MB
alpine          latest   8ca4688f4f35  3 weeks ago  7.34MB
hello-world     latest   9c7a54a9a43c  5 months ago  13.3kB
hello-world     linux    9c7a54a9a43c  5 months ago  13.3kB
linuxconfig/lemp latest   670ff9140a94  13 months ago 1.05GB
[kralos]--[✗ main = ● ]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\nodejs\dockerfiles]
▶ █
```

Dockerfile {ejemplo}



The screenshot shows the VS Code interface with the following details:

- EXPLORER:** NOTAS_TI1
- dockerfile** (active tab)
- Content:**

```
FROM alpine
CMD ["/bin/echo", "Hi Docker. I'm a dockerfile !"]
```
- PROBLEMS:** No problems found.
- OUTPUT:**
 - [[kralos]--[?]main = ●]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\nodejs\dockerfiles]
 - ► docker build -t hola-mundo .
[+] Building 3.0s (5/5) FINISHED
=> [internal] load .dockerignore
=> => transferring context: 2B
● => [internal] load build definition from dockerfile
● => => transferring dockerfile: 103B
=> [internal] load metadata for docker.io/library/alpine:latest
=> CACHED [1/1] FROM docker.io/library/alpine
● => exporting to image
● => => exporting layers
=> => writing image sha256:8e2d75649edb4f0b043a448ad474a82bcb49278092ad5cd74d8db55f04b2e4db
● => => naming to docker.io/library/hola-mundo
 - What's Next?
View a summary of image vulnerabilities and recommendations → docker scout quickview
 - [[kralos]--[?]main = ●]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\nodejs\dockerfiles]
 - ► docker run hola-mundo
● Hi Docker. I'm a dockerfile !
 - [[kralos]--[?]main = ●]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\nodejs\dockerfiles]
 - ► █
- DEBUG CONSOLE, TERMINAL, PORTS, GITLENS:** Not visible in the screenshot.

Dockerfile {ejemplo}

```
● ▶ docker run hola-mundo
● Hi Docker. I'm a dockerfile !
[ kralos ]--[ \main = ● ]
● [D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\nodejs\dockerfiles]
● ▶ docker images
● REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
● mongo          latest    ee3b4d1239f1  7 days ago   748MB
● alpine         latest    8ca4688f4f35  3 weeks ago  7.34MB
● hola-mundo     latest    8e2d75649edb  3 weeks ago  7.34MB
● hello-world    latest    9c7a54a9a43c  5 months ago 13.3kB
● hello-world    linux     9c7a54a9a43c  5 months ago 13.3kB
● linuxconfig/lemp  latest   670ff9140a94  13 months ago 1.05GB
○ [ kralos ]--[ \main = ● ]
○ [D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\nodejs\dockerfiles]
○ ▶ docker ps -a
● CONTAINER ID  IMAGE      COMMAND      CREATED      STATUS      PORTS      NAMES
● 071fed489ee  hola-mundo  "/bin/echo 'Hi Docke..."  2 minutes ago  Exited (0) About a minute ago
● 29293400974a  mongo      "docker-entrypoint.s..."  25 hours ago  Exited (0) 24 hours ago
● f3c176d990da  linuxconfig/lemp  "supervisord"      2 days ago   Exited (137) 2 days ago
● 140a76a8ce6a  linuxconfig/lemp  "supervisord"      2 weeks ago  Exited (137) 2 days ago
● [ kralos ]--[ \main = ● ]
● [D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\nodejs\dockerfiles]
○ ▶ █
```

Actualización de imágenes docker

```
[kralos]--[¶ ≠]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1]
● ▶ docker images
REPOSITORY          TAG      IMAGE ID      CREATED       SIZE
node                latest   d7eb1d080096  3 days ago   1.1GB
mongo               latest   ee3b4d1239f1  10 days ago  748MB
alpine              latest   8ca4688f4f35  3 weeks ago  7.34MB
hola-mundo         latest   8e2d75649edb  3 weeks ago  7.34MB
hello-world         latest   9c7a54a9a43c  5 months ago 13.3kB
hello-world         linux    9c7a54a9a43c  5 months ago 13.3kB
linuxconfig/lEMP   latest   670ff9140a94  13 months ago 1.05GB
[kralos]--[¶main ≡ ● ]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1]
○ ▶ █
```

Practica 4

The screenshot shows the VS Code interface with the following details:

- EXPLORER:** NOTAS_TI1
- File Tree (left):** code\Nodejs, dockerfiles, proyecto_0, proyecto_1, proyecto_2 (expanded), css, js, Dockerfile, index.html
- Editor (right):** Dockerfile (selected)
- Content of Dockerfile:**

```
1 FROM linuxconfig/lEMP
2
3 RUN mkdir -p /var/www/html/mysite
4
5 COPY . /var/www/html/mysite
```
- Bottom Navigation Bar:** PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, PORTS, GITLENS

Practica 4

```
[kralos]--[¶]main = ● ]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1]
▶ cd .\code\nodejs\proyecto_2\
[kralos]--[¶]main = ● ]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\nodejs\proyecto_2]
● ▶ ls

Directory: D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\nodejs\proyecto_2

      Mode          LastWriteTime      Length Name
-----  -----          10/23/2023 11:07 AM           ■  css
      Mode          LastWriteTime      Length Name
-----  -----          10/23/2023 11:07 AM           ■  js
-a---  Mode          LastWriteTime      Length Name
-----  -----          10/23/2023 11:41 AM           89  ◆ Dockerfile
-a---  Mode          LastWriteTime      Length Name
-----  -----          10/23/2023 11:10 AM          346  ◇ index.html

[kralos]--[¶]main = ● ]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\nodejs\proyecto_2]
● ▶ docker build -t mysite:0.0.1 .
[+] Building 0.0s (0/0)
[+] Building 46.3s (8/8) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 126B
=> [internal] load .dockerrcignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/linuxconfig/lEMP:latest
=> [1/3] FROM docker.io/linuxconfig/lEMP
=> [internal] load build context
=> => transferring context: 623B
=> [2/3] RUN mkdir -p /var/www/html/mysite
=> [3/3] COPY . /var/www/html/mysite
=> exporting to image
=> => exporting layers
=> => writing image sha256:e8a126f7e3b911f911b32daf95225f2bcf3c6d07ea296a78873681f17e0d675d
=> => naming to docker.io/library/mysite:0.0.1

What's Next?
View a summary of image vulnerabilities and recommendations → docker scout quickview
[kralos]--[¶]main = ● ]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\nodejs\proyecto_2]
● ▶ █
```

Practica 4

```
[kralos]--[¶ ≠]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\nodejs\proyecto_2]
● ► docker images
REPOSITORY          TAG      IMAGE ID      CREATED       SIZE
mysite              0.0.1    e8a126f7e3b9  15 minutes ago  1.05GB
● node               latest   d7eb1d080096  3 days ago    1.1GB
mongo               latest   ee3b4d1239f1  10 days ago   748MB
alpine              latest   8ca4688f4f35  3 weeks ago   7.34MB
hola-mundo         latest   8e2d75649edb  3 weeks ago   7.34MB
hello-world         latest   9c7a54a9a43c  5 months ago  13.3kB
hello-world         linux    9c7a54a9a43c  5 months ago  13.3kB
linuxconfig/lemp   latest   670ff9140a94  13 months ago 1.05GB
[kralos]--[¶main = ● ]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\nodejs\proyecto_2]
○ ► █
```

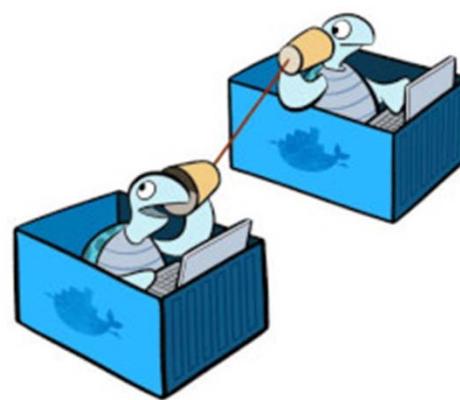
Practica 4

```
[kralos]--[?]main = ● ][x125]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\nodejs\proyecto_2]
● ▶ docker run --name=mi_sitio -dP mysite:0.0.1
f05bd8066b4e5a56c691c3f961cacc8b103975608ebd2a0eda57d80a7589743e
[kralos]--[?]main = ● ]
● [D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\nodejs\proyecto_2]
● ▶ docker port mi_sitio
80/tcp -> 0.0.0.0:32769
○ 3306/tcp -> 0.0.0.0:32768
[kralos]--[?]main = ● ]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\nodejs\proyecto_2]
○ ▶ docker exec -it mi_sitio /bin/bash
root@f05bd8066b4e:# ls /var/www/html/
index.nginx-debian.html index.php           mysite/
root@f05bd8066b4e:# ls /var/www/html/mysite/
Dockerfile css index.html js
root@f05bd8066b4e:# cat /var/www/html/mysite/index.html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>In_Docker</title>
    <link rel="stylesheet" href=".css/style.css">
  </head>
  <body>
    <h1>Hola estoy en un contendor ...</h1>
  </body>
  <script src=".js/main.js"></script>
</html>root@f05bd8066b4e:# 
```



The terminal window shows the steps to run a Docker container named 'mi_sitio' with port mapping. It then lists the ports exposed (80/tcp and 3306/tcp) and enters the container via 'docker exec'. Inside the container, it lists the files in the '/var/www/html/' directory, including 'index.nginx-debian.html', 'index.php', and a folder 'mysite' containing 'Dockerfile', 'css', 'index.html', and 'js'. It then displays the contents of 'index.html' which contains the text 'Hola estoy en un contendor ...'. The browser window titled 'In_Docker' shows the same text 'Hola estoy en un contendor ...' at the URL 'localhost:32769/mysite/index.html'.

Redes en Docker



Redes en Docker

- Por defecto los contenedores que creamos se conectan a la red de tipo bridge llamada **bridge** (por defecto el direccionamiento de esta red es 172.17.0.0/16). Los contenedores conectados a esta red que quieren exponer algún puerto al exterior tienen que usar la opción -p para mapear puertos.

```
▶ docker network ls
NETWORK ID      NAME      DRIVER      SCOPE
c42d81defb27    bridge    bridge      local
6ae7b1eda800    host      host       local
193e4faae928    none     null       local
```

Redes en Docker

```
[kralos]
[~]
▶ docker network inspect bridge
[
  {
    "Name": "bridge",
    "Id": "c42d81defb27afc6f15350402006b61fd2bd7d529e4e42d2c4978b69e80e577a",
    "Created": "2023-11-06T17:00:53.246Z",
    "Scope": "local",
    "Driver": "bridge",
    "EnableIPv6": false,
    "IPAM": {
      "Driver": "default",
      "Options": null,
      "Config": [
        {
          "Subnet": "172.17.0.0/16",
          "Gateway": "172.17.0.1"
        }
      ]
    },
    "Internal": false,
    "Attachable": false,
    "Ingress": false,
    "ConfigFrom": {
      "Network": ""
    },
    "ConfigOnly": false,
    "Containers": {},
    "Options": {
      "com.docker.network.bridge.default_bridge": "true",
      "com.docker.network.bridge.enable_icc": "true",
      "com.docker.network.bridge.enable_ip_masquerade": "true",
      "com.docker.network.bridge.host_binding_ipv4": "0.0.0.0",
      "com.docker.network.bridge.name": "docker0",
      "com.docker.network.driver.mtu": "1500"
    },
    "Labels": {}
  }
]
```

Redes en Docker

--network nombre_red

- Para conectar un contenedor a una red.
- Si conecto un contenedor a la red **host**, el contenedor será accesible usando la misma IP que tu máquina.
- La red **none** no configurará ninguna IP para el contenedor y no tiene acceso a la red externa ni a otros contenedores. Tiene la dirección loopback y se puede usar para ejecutar trabajos por lotes.

Redes en Docker

- Crea una red dentro de Docker

```
docker network create nombre-red
```

- Crea una red con rango dentro de Docker

```
docker network create -d macvlan \  
--subnet=172.16.86.0/24 \  
--gateway=172.16.86.1 \  
-o parent=eth0 name-red
```

- Borrar una red dentro de Docker

```
docker network rm nombre-red
```

Redes en Docker

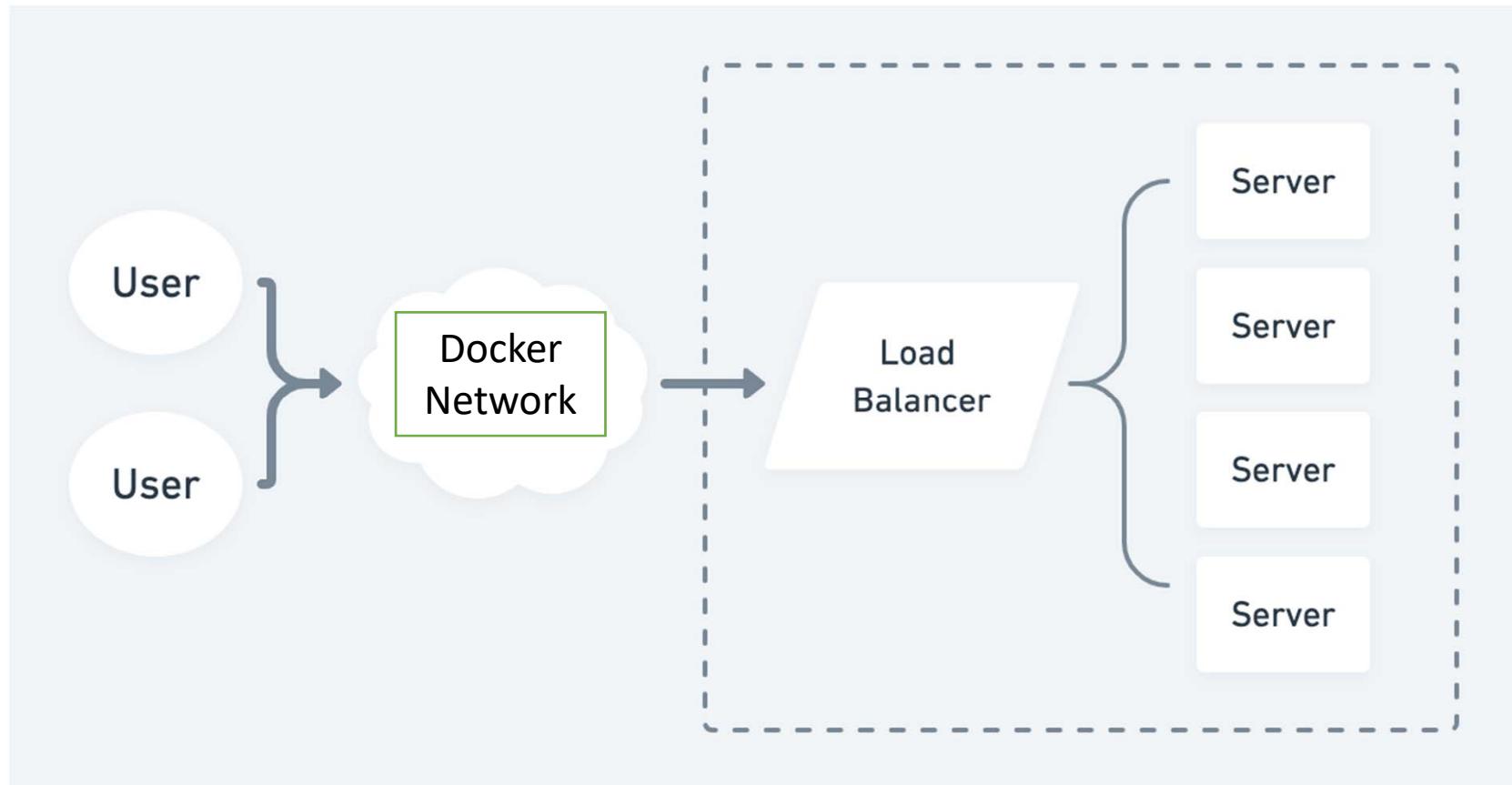
```
▶ docker network create -d macvlan --subnet=172.16.86.0/24 --gateway=172.16.86.1 -o parent=eth0 my-macvlan-net
6f61c1ecf426d808f39be8436de6c2de4e868f0f80eb3776acdcb4f4471dfaf1
[kralos]
[~]
▶ docker network ls
NETWORK ID      NAME      DRIVER      SCOPE
c42d81defb27   bridge    bridge      local
6ae7b1eda800   host      host       local
6f61c1ecf426   my-macvlan-net  macvlan   local
193ed4faae928  none      null      local
[kralos]
[~]
▶ docker network inspect my-macvlan-net
[
  {
    "Name": "my-macvlan-net",
    "Id": "6f61c1ecf426d808f39be8436de6c2de4e868f0f80eb3776acdcb4f4471dfaf1",
    "Created": "2023-11-06T17:32:47.8301725Z",
    "Scope": "local",
    "Driver": "macvlan",
    "EnableIPv6": false,
    "IPAM": {
      "Driver": "default",
      "Options": {},
      "Config": [
        {
          "Subnet": "172.16.86.0/24",
          "Gateway": "172.16.86.1"
        }
      ]
    },
    "Internal": false,
    "Attachable": false,
    "Ingress": false,
    "ConfigFrom": {
      "Network": ""
    },
    "ConfigOnly": false,
    "Containers": {},
    "Options": {
      "parent": "eth0"
    },
    "Labels": {}
  }
]
```

Practica 5

- Enrutamiento y Balanceo

```
[kralos]--[¶main ≡ ● ]  
[[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1]]  
● ► docker network create mynet  
36367c36d4688ca5153e9562adf9edfc4e79002589df15655b311139f738e29  
[[kralos]--[¶main ≡ ● ]  
[[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1]]  
● ► docker network ls  
NETWORK ID      NAME      DRIVER      SCOPE  
500fcdf0c8d4   bridge    bridge      local  
6ae7b1eda800   host      host       local  
6f61c1ecf426   my-macvlan-net macvlan   local  
36367c36d468   mynet     bridge      local  
193e4faae928   none      null       local  
[[kralos]--[¶main ≡ ● ]  
[[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1]]  
● ► docker images  
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE  
nginx           mainline-alpine3.18-slim ab7c6a6862ef  2 weeks ago  17MB  
nginx           latest    593aeee2afb64  2 weeks ago  187MB  
mysite          0.0.1    e8a126f7e3b9  2 weeks ago  1.05GB  
node            latest    d7eb1d080096  2 weeks ago  1.1GB  
mongo           latest    ee3b4d1239f1  3 weeks ago  748MB  
alpine          latest    8ca4688f4f35  5 weeks ago  7.34MB  
holamundo       latest    8e2d75649edb  5 weeks ago  7.34MB  
hello-world     latest    9c7a54a9a43c  6 months ago 13.3kB  
hello-world     linux     9c7a54a9a43c  6 months ago 13.3kB  
linuxconfig/lemp  latest    670ff9140a94  14 months ago 1.05GB
```

Practica 5



Practica 5

```
[kralos]--[¶]main = ● ][x125]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\networking\Balanceo]
● ► docker network inspect mynet
[
  {
    "Name": "mynet",
    "Id": "36367c36d4688ca5153e9562adf9edfc4e79002589df15655b311139f738e29",
    "Created": "2023-11-08T16:44:07.691363211Z",
    "Scope": "local",
    "Driver": "bridge",
    "EnableIPv6": false,
    "IPAM": {
      "Driver": "default",
      "Options": {},
      "Config": [
        {
          "Subnet": "172.18.0.0/16",
          "Gateway": "172.18.0.1"
        }
      ]
    },
    "Internal": false,
    "Attachable": false,
    "Ingress": false,
    "ConfigFrom": {
      "Network": ""
    },
    "ConfigOnly": false,
    "Containers": {},
    "Options": {},
    "Labels": {}
  }
]
```

Practica 5

The screenshot shows a Docker interface with two entries:

- nginx** (latest): 7f553e8bbc89. Status: Unused. Created 14 days ago. Size: 191.67 MB.
- nginx** (mainline-alpine3.20-slim): d769f03e916d. Status: Unused. Created 14 days ago. Size: 11.79 MB.

Below the Docker interface is a code editor window with three tabs:

- index1.html**: Contains the following code:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Server 1</title>
</head>
<body>
<p>Soy el server 1</p>
</body>
</html>
```
- index2.html**: Contains the following code:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Server 2</title>
</head>
<body>
<p>Soy el server 2</p>
</body>
</html>
```
- index3.html**: Contains the following code:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Server 3</title>
</head>
<body>
<p>Soy el server 3</p>
</body>
</html>
```

On the left, there is an **EXPLORER** sidebar showing the file structure for each container, including **default.conf** and **index1.html**, **index2.html**, and **index3.html**.

Below the code editor is another code editor window with one tab:

- default.conf**: Contains the following configuration:

```
upstream myapp1 {
    server 172.18.0.2;
    server 172.18.0.3;
    server 172.18.0.4;
}

server {
    listen 80;
    location / {
        proxy_pass http://myapp1;
    }
}
```

On the left, there is an **EXPLORER** sidebar showing the file structure for this configuration, including **default.conf** and **index1.html**, **index2.html**, and **index3.html**.

Practica 5

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

[[kralos]
[D:\htmls]
● ▶ docker run --rm -dP -v D:\htmls\index1.html:/usr/share/nginx/html/index.html --network mynet nginx d88a5df75f202e43bc4ef07b81bd95c89acf11d1a0fb6472b3e8ba51673e8ddf
[[kralos]
[D:\htmls]
● ▶ docker run --rm -dP -v D:\htmls\index2.html:/usr/share/nginx/html/index.html --network mynet nginx 470d8a1f717e235522069f9cdd607853285b0cbb8447f3bfbabf23d056156973
[[kralos]
[D:\htmls]
● ▶ docker run --rm -dP -v D:\htmls\index3.html:/usr/share/nginx/html/index.html --network mynet nginx 0e4809f540de530247d47ad1ef75c5e4d10a5ba2fbdfb06fdca90fd86b839f24
○ [[kralos]
[D:\htmls]
▶ ]]
```

Practica 5

The screenshot shows the Docker UI interface. On the left, there is a sidebar with the following options:

- Containers (selected)
- Images
- Volumes
- Builds
- Docker Scout
- Extensions

The main area is titled "Containers" with a "Give feedback" link. It displays performance metrics: Container CPU usage (0.00% / 800%) and Container memory usage (0B / 6.58GB). There is also a "Show charts" button.

Below the metrics, there is a search bar and a filter toggle labeled "Only show running containers".

A table lists three running containers:

	Name	Image	Status	Port(s)	CPU (%)	Last started	Actions
<input type="checkbox"/>	trusting_le d88a5df75f	nginx:<none>	Running	32768:80 ↗	0%	43 seconds ago	[Actions]
<input type="checkbox"/>	musing_jo 470d8a1f71	nginx:<none>	Running	32769:80 ↗	0%	38 seconds ago	[Actions]
<input type="checkbox"/>	jolly_dijkstra 0e4809f540	nginx:<none>	Running	32770:80 ↗	0%	34 seconds ago	[Actions]

Practica 5

trusting_leakey

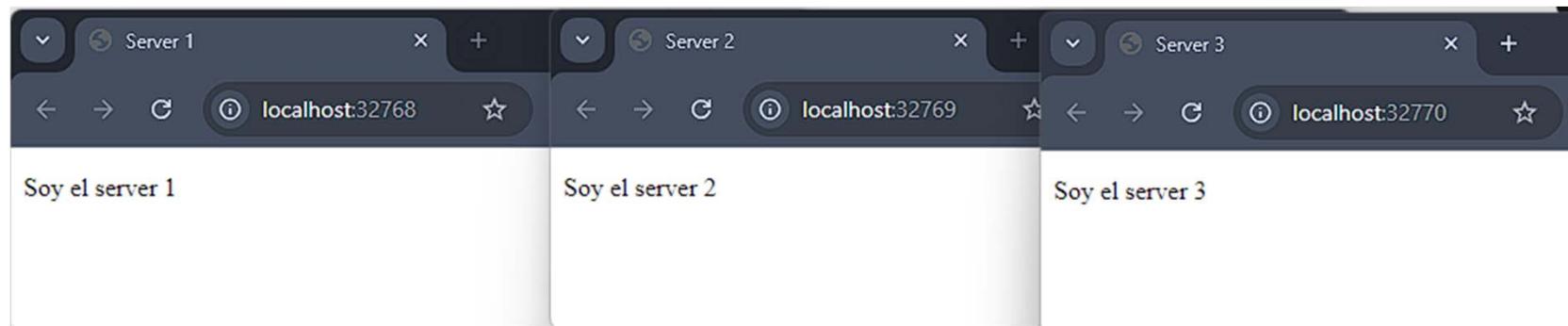
```

<  d88a5df75f20 ⚡  nginx:latest (was nginx:<none>) <  470d8a1f717e ⚡  nginx:latest (was nginx:<none>) <  0e4809f540de ⚡  nginx:latest (was nginx:<none>)
32768:80 ↗ 32769:80 ↗ 32770:80 ↗

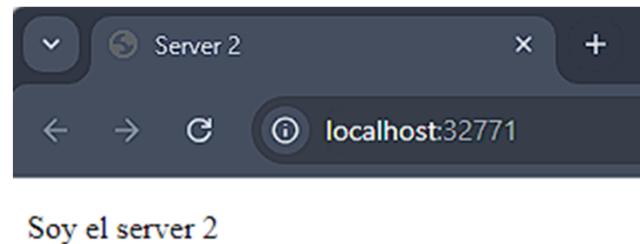
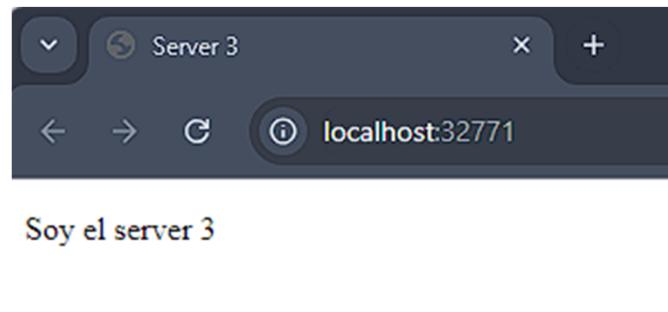
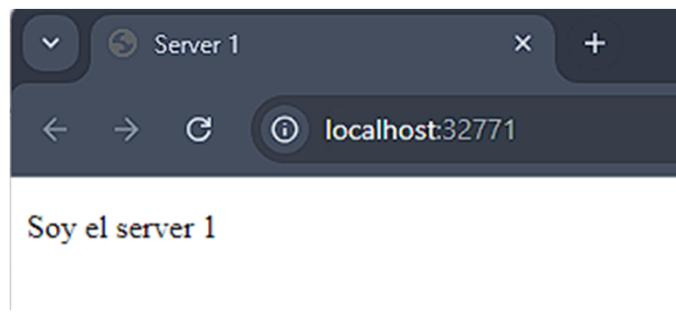
```

Logs	Inspect	Bind mounts	Exec	Files	Stats	Logs	Inspect	Bind mounts	Exec	Files	Stats	Logs	Inspect	Bind mounts	Exec	Files	Stats
Platform	Cmd	State	Image	PortBindings		Platform	Cmd	State	Image	PortBindings		Platform	Cmd	State	Image	PortBindings	
Networks						Networks						Networks					
212 v	"Networks": {		212 v	"Networks": {		212 v	"Networks": {		212 v	"Networks": {		212 v	"Networks": {		212 v	"Networks": {	
213 v	"mynet": {		213 v	"mynet": {		213 v	"mynet": {		213 v	"mynet": {		213 v	"mynet": {		213 v	"mynet": {	
214	"IPAMConfig": null,		214	"IPAMConfig": null,		214	"IPAMConfig": null,		214	"IPAMConfig": null,		214	"IPAMConfig": null,		214	"IPAMConfig": null,	
215	"Links": null,		215	"Links": null,		215	"Links": null,		215	"Links": null,		215	"Links": null,		215	"Links": null,	
216	"Aliases": null,		216	"Aliases": null,		216	"Aliases": null,		216	"Aliases": null,		216	"Aliases": null,		216	"Aliases": null,	
217	"MacAddress": "02:42:ac:12:00:02"		217	"MacAddress": "02:42:ac:12:00:03"		217	"MacAddress": "02:42:ac:12:00:03"		217	"MacAddress": "02:42:ac:12:00:04"		217	"MacAddress": "02:42:ac:12:00:04"		217	"MacAddress": "02:42:ac:12:00:04"	
218	"DriverOpts": null,		218	"DriverOpts": null,		218	"DriverOpts": null,		218	"DriverOpts": null,		218	"DriverOpts": null,		218	"DriverOpts": null,	
219	"NetworkID": "180a4a11473237c41b7"		219	"NetworkID": "180a4a11473237c41b7"		219	"NetworkID": "180a4a11473237c41b7"		219	"NetworkID": "180a4a11473237c41b7"		219	"NetworkID": "180a4a11473237c41b7"		219	"NetworkID": "180a4a11473237c41b7"	
220	"EndpointID": "b21c721e6a96f8f69c"		220	"EndpointID": "4add9f6e576301e370"		220	"EndpointID": "4add9f6e576301e370"		220	"EndpointID": "879029b0dd1767277c"		220	"EndpointID": "879029b0dd1767277c"		220	"EndpointID": "879029b0dd1767277c"	
221	"Gateway": "172.18.0.1",		221	"Gateway": "172.18.0.1",		221	"Gateway": "172.18.0.1",		221	"Gateway": "172.18.0.1",		221	"Gateway": "172.18.0.1",		221	"Gateway": "172.18.0.1",	
222	"IPAddress": "172.18.0.2",		222	"IPAddress": "172.18.0.3",		222	"IPAddress": "172.18.0.4",		222	"IPAddress": "172.18.0.4",		222	"IPAddress": "172.18.0.4",		222	"IPAddress": "172.18.0.4",	
223	"IPPrefixLen": 16,		223	"IPPrefixLen": 16,		223	"IPPrefixLen": 16,		223	"IPPrefixLen": 16,		223	"IPPrefixLen": 16,		223	"IPPrefixLen": 16,	
224	"IPv6Gateway": "",		224	"IPv6Gateway": "",		224	"IPv6Gateway": "",		224	"IPv6Gateway": "",		224	"IPv6Gateway": "",		224	"IPv6Gateway": "",	
225	"GlobalIPv6Address": "",		225	"GlobalIPv6Address": "",		225	"GlobalIPv6Address": "",		225	"GlobalIPv6Address": "",		225	"GlobalIPv6Address": "",		225	"GlobalIPv6Address": "",	
226	"GlobalIPv6PrefixLen": 0,		226	"GlobalIPv6PrefixLen": 0,		226	"GlobalIPv6PrefixLen": 0,		226	"GlobalIPv6PrefixLen": 0,		226	"GlobalIPv6PrefixLen": 0,		226	"GlobalIPv6PrefixLen": 0,	
227 v	"DNSNames": [227 v	"DNSNames": [227 v	"DNSNames": [227 v	"DNSNames": [227 v	"DNSNames": [227 v	"DNSNames": [
228	"trusting_leakey",		228	"musing_johnson",		228	"jolly_dijkstra",		228	"470d8a1f717e"		228	"0e4809f540de"		228	"0e4809f540de"	
229	"d88a5df75f20"		229			229			229			229			229		
230]		230			230			230			230			230		
231	}		231			231			231			231			231		

Practica 5



Practica 5

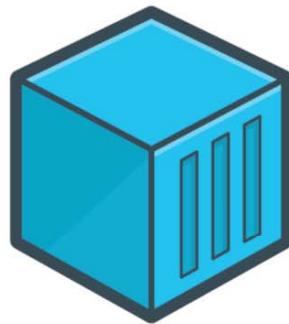


YAML

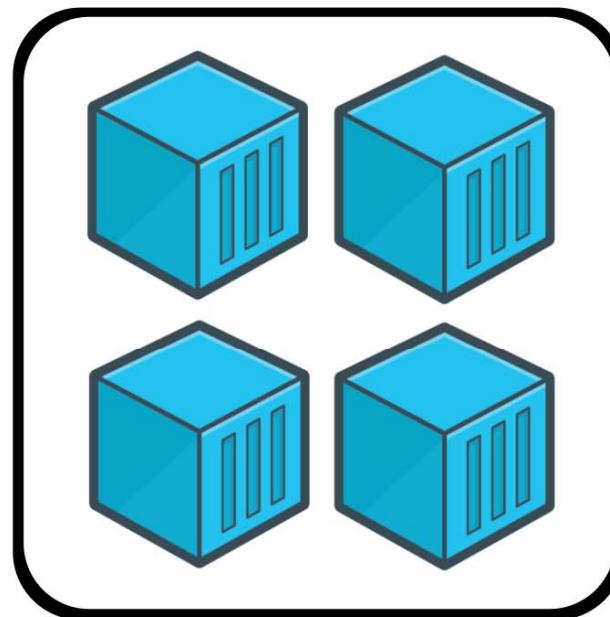
{YAML Ain't Markup Language}

- YAML es un lenguaje de serialización de datos y suele utilizarse en el diseño de archivos de configuración, no para los documentos.
- YAML utiliza una extensión de archivos .yml o .yaml y sigue reglas de sintaxis específicas.
- Tiene características que provienen de Perl, C, XML, HTML y otros lenguajes de programación. También se basa en JSON, por lo que los archivos JSON son compatibles con YAML.

Docke vs Docker Compose



Docker



Docker Compose

Docker Compose

- Es una herramienta de Docker que orquesta contenedores en un mismo cliente. Consiste en un archivo de texto en formato YAML, que define de forma declarativa los contenedores que se van a desplegar, así como las dependencias entre ellos.
- Este archivo normalmente tiene el nombre docker-compose.yml pero puede tener cualquier otro nombre, siempre que se especifique al ejecutar el comando docker-compose con el argumento -f. La utilidad más habitual para esta herramienta es el despliegue de aplicaciones en entornos locales, para el desarrollo y pruebas.

Docker Compose

```
[kralos]--[|main ≡]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1]
● ▶ docker-compose --version
Docker Compose version v2.23.0-desktop.1
```

The screenshot shows a code editor interface with three tabs open:

- Dockerfile**:

```
FROM node:20.9.0-alpine3.18
WORKDIR /home/nodeApp
EXPOSE 8080
CMD ["node","index.js"]
```
- docker-compose.yml**:

```
version: '2.23'
services:
  web:
    container_name: nodeDockerApp
    build: .
  ports:
    - "8080:8080"
  volumes:
    - ./nodeApp:/home/nodeApp
```
- index.js**:

```
console.log("Hi, docker!!!");
```

The **index.js** file is currently selected in the Explorer sidebar, which also lists other projects like `balanceo`, `node-whatsapp-apirest`, `nodejs`, and `rasa`.

Docker Compose

```
[kralos]--[¶]main ≡ ● ][x17]
● [[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\docker-composer\ejemplo_0]
▶ docker-compose build
[+] Building 16.2s (6/6) FINISHED
  => [web internal] load build definition from Dockerfile
  => => transferring dockerfile: 125B
  => [web internal] load .dockerignore
  => => transferring context: 2B
  => [web internal] load metadata for docker.io/library/node:20.9.0-alpine3.18
  => [web 1/2] FROM docker.io/library/node:20.9.0-alpine3.18
● => [web 2/2] WORKDIR /home/nodeApp
  => [web] exporting to image
○ => => exporting layers
● => => writing image sha256:1f33cdcf452c002edc6abb0c7df55ee730bae82463f4bf682807c7ae22a5b33
  => => naming to docker.io/library/ejemplo_0-web
[[kralos]--[¶]≡]
● [[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\docker-composer\ejemplo_0]
▶ docker-compose up
[+] Building 0.0s (0/0)
[+] Running 2/2
  ✓ Network ejemplo_0_default  Created
  ✓ Container nodeDockerApp  Created
Attaching to nodeDockerApp
nodeDockerApp  | Hi, docker!!!
nodeDockerApp exited with code 0
[[kralos]--[¶]≡]
● [[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\docker-composer\ejemplo_0]
▶ █
```

Docker Compose

- Lo primero es definir los servicios que vamos a desplegar.
- Estos servicios son los que conforman nuestra aplicación y, normalmente, cada uno despliega un contenedor, asociado a una imagen Docker. Dentro de cada servicio podemos definir las características de cada contenedor, como el nombre, la imagen, los puertos que expone, volúmenes y redes a las que se conecta, etc.

Docker Compose

- version “2.23”: La versión de Docker Compose.
- services: Definimos los contenedores que se crearan. Por ejemplo, Services, web y database.
- web: El nombre de un service.
- build: La ubicación del Dockerfile.
- ports: El mapeo de puertos.
- volumes: Directorios a compartir con él o los contenedores.
- networks: Red a la que pertenecerá el servicio

Referencias

- *Comandos más usados en Docker.* (n.d.). Tutoriales.online. Retrieved October 4, 2023, from <https://tutoriales.online/chuletas/docker>
- *Docker.* (n.d.). Docker.com. Retrieved October 4, 2023, from <https://hub.docker.com/>
- *Docker.* (n.d.). Docker.com. Retrieved October 4, 2023, from <https://hub.docker.com/>
- Avi. (2023, September 14). *¿Qué es Dockerfile y cómo crear una imagen Docker?* Geekflare. <https://geekflare.com/es/dockerfile-tutorial/>
- Muñoz, J. D. (2020, February 18). *Introducción a las redes en docker. Enlazando contenedores docker.* PLEDIN 3.0. <https://www.josedomingo.org/pledin/2020/02/redes-en-docker/>
- Torres, G. (2020, December 23). *Cómo funcionan las redes en Docker - return(GiS); return(GiS); Gisela Torres.* <https://www.returngis.net/2020/12/como-funcionan-las-redes-en-docker/>
- *¿Qué es YAML?* (n.d.). Redhat.com. Retrieved November 6, 2023, from <https://www.redhat.com/es/topics/automation/what-is-yaml>
- *Docker Compose Tutorial: advanced Docker made simple.* (n.d.). Eduative. Retrieved November 13, 2023, from <https://www.educative.io/blog/docker-compose-tutorial>
- Urbina, M. S. (2020, April 21). *Docker-compose con ejemplos - Mario Sanchez Urbina.* Medium. <https://mariosanurbina.medium.com/docker-compose-con-ejemplos-1290c0a3ae53>

Referencias {Código}

- <https://github.com/nschurmann/mongoapp-curso-docker>