

Bloque 1 -Tema 4. Docker.

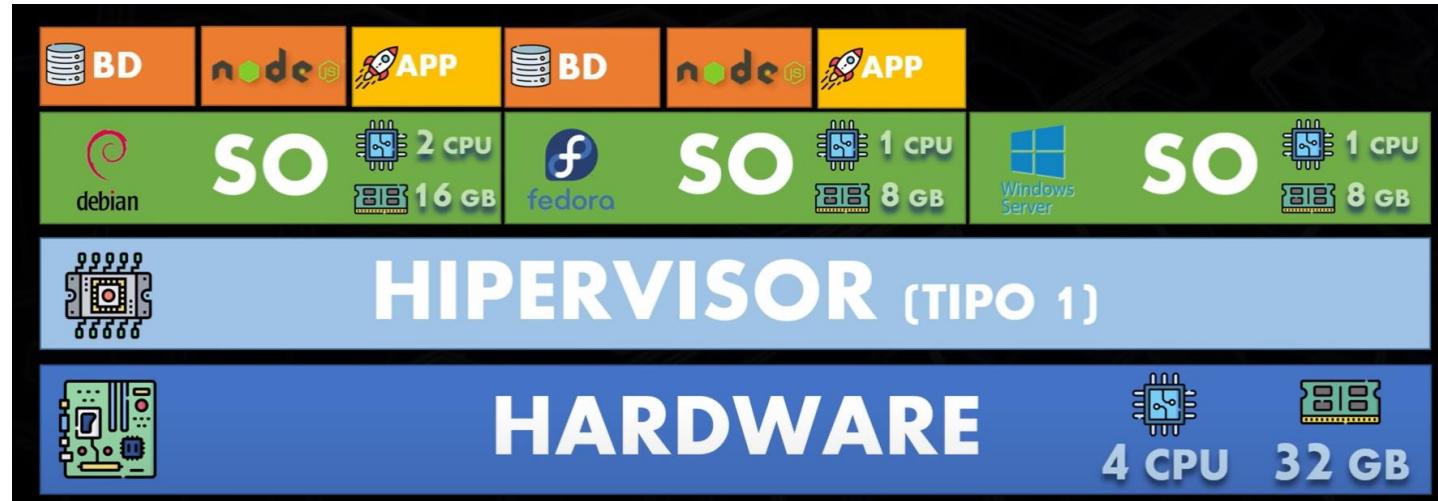
Contenedores para la Gestión, Orquestación y
Procesamiento Escalable de Datos en entornos
Big Data y Data Engineering

Servidor físico vs máquina virtual



Servidor
Físico

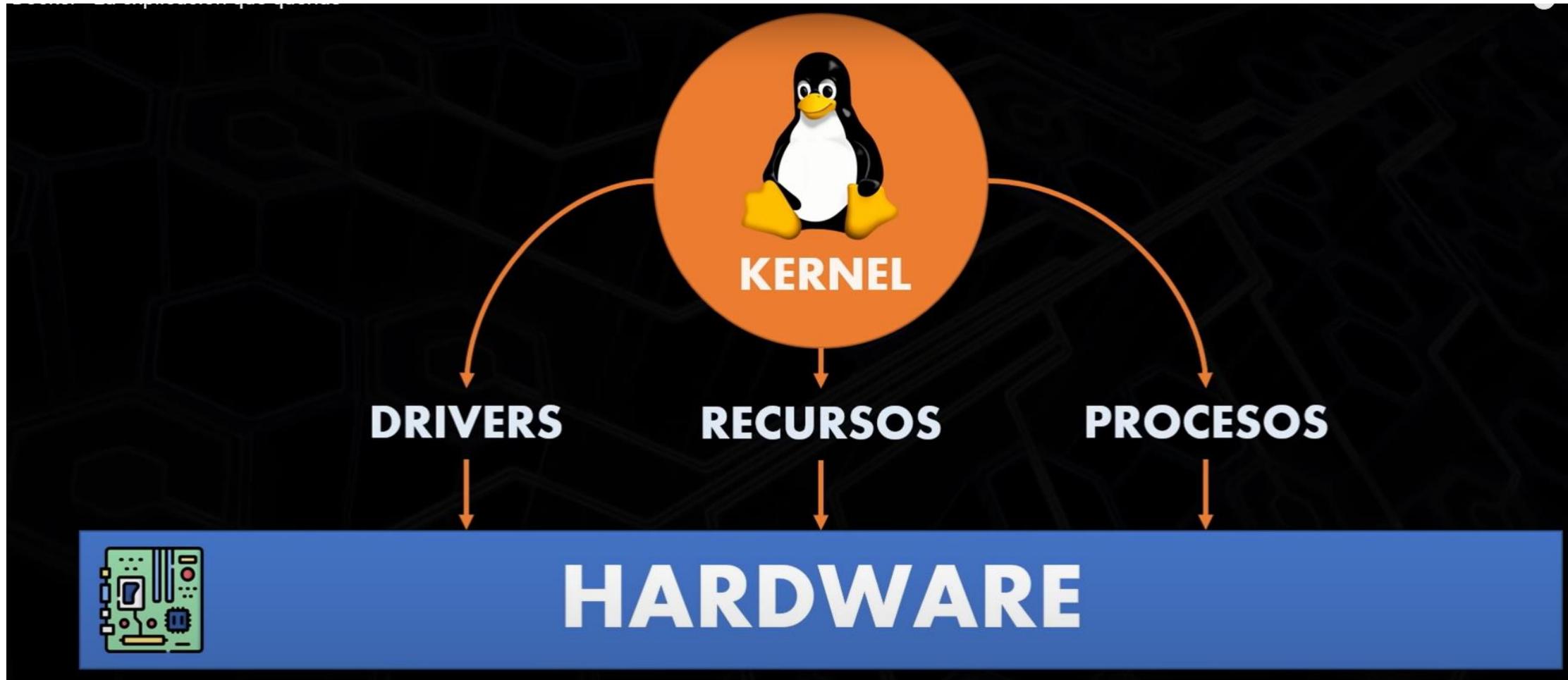
Máquina
Virtual



Al aumentar el nro de máquinas virtuales se incrementa el costo en hardware.

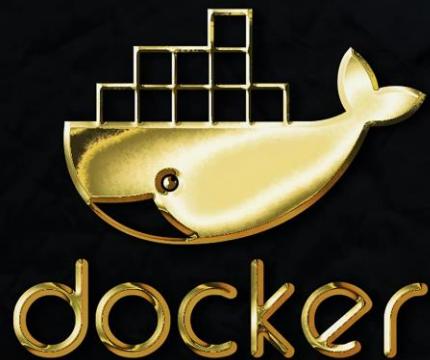


En vez de tener distintos kernels ¿Por qué no utilizar un único kernel a partir del cual se gestionen todos los servicios?



En vez de tener distintos kernels
¿Por qué no utilizar un único kernel a partir del cual se
gestionen todos los servicios?





Docker es un conjunto de herramientas que se utiliza para empaquetar aplicaciones con todas sus herramientas y bibliotecas necesarias en "contenedores" ordenados y portátiles. Estos contenedores se ejecutan en cualquier lugar, aislados unos de otros, lo que garantiza un rendimiento uniforme y eficiente.

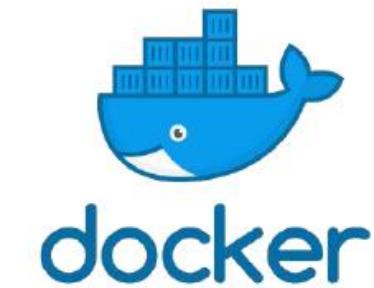
Contenedores y contenedorización (Containers and Containerization)

Un contenedor es como un conjunto de bibliotecas, dependencias y archivos de configuración. En comparación con las máquinas virtuales que emulan sistemas operativos completos, los contenedores comparten el núcleo (kernel) del host, lo que los hace más ligeros y rápidos.

Cada contenedor está aislado del otro, por lo que es independiente y sus procesos y recursos no interfieren entre sí.

¿Qué son contenedores?

Los contenedores son un paquete de elementos que permiten ejecutar una aplicación determinada en cualquier sistema operativo.



Docker

Linux-VServer

LXC

LXD

OpenVZ

Systemd-nspawn

Contenedores y contenedorización (Containers and Containerization)

La contenedorización es el proceso de empaquetar una aplicación en un contenedor estandarizado. Docker es una popular plataforma de contenedorización que proporciona herramientas para crear, compartir y ejecutar contenedores.

La contenedorización beneficia a la ciencia de datos de muchas maneras:

Containerization

Reproducibilidad: Ejecute su análisis con el mismo entorno (versiones de software, dependencias) en cualquier lugar, garantizando la coherencia de los resultados.

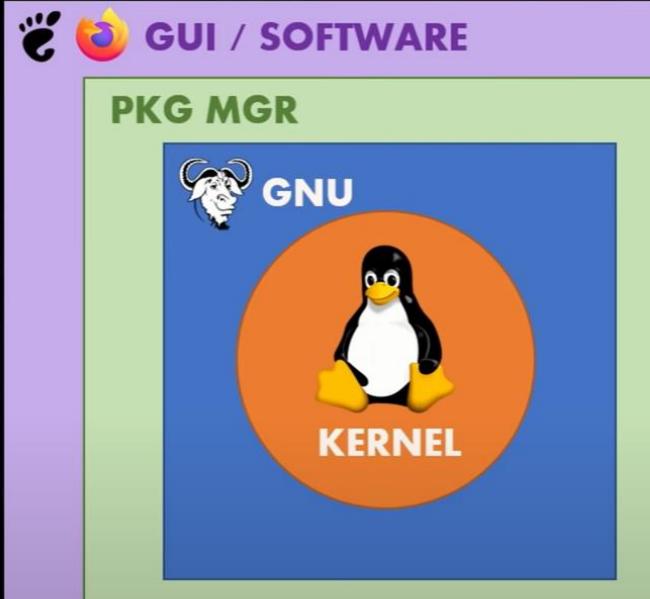
Portabilidad: Comparta y traslade fácilmente sus proyectos de ciencia de datos entre máquinas sin preocuparse por la configuración del entorno.

Eficiencia de recursos: Múltiples contenedores pueden ejecutarse en una sola máquina, maximizando los recursos informáticos para el procesamiento y análisis de datos.

Aislamiento: Aíslle tareas específicas como la ingeniería de características o la formación de modelos para una mejor colaboración y experimentación.

Máquinas virtuales frente a Docker

Máquinas virtuales



Contenedores



Máquinas virtuales frente a Docker

Las máquinas virtuales (VM) y los contenedores Docker son tecnologías utilizadas para aislar entornos de software, pero difieren significativamente en su enfoque y casos de uso. Comprender estas diferencias es crucial para elegir la herramienta adecuada para sus necesidades específicas.

Máquinas virtuales (VMs):

Emular un sistema operativo completo: Cada VM actúa como un ordenador virtual con su CPU, memoria, almacenamiento y sistema operativo.

Pros:

Aislamiento total: Las máquinas virtuales ofrecen un fuerte aislamiento entre entornos, lo que las hace ideales para ejecutar aplicaciones con dependencias conflictivas o requisitos de seguridad.

Máquinas virtuales (VMs):

Pros:

Flexibilidad: Las máquinas virtuales pueden ejecutar cualquier sistema operativo, lo que permite crear entornos adaptados a necesidades específicas.

Máquinas virtuales (VMs):

Contras:

Consumo intensivo de recursos: Las máquinas virtuales requieren muchos recursos, lo que limita el número de ellas que se pueden ejecutar en una sola máquina.

Arranque lento: Arrancar un sistema operativo completo lleva tiempo, por lo que las máquinas virtuales tardan más en iniciarse que los contenedores.

Contenedores Docker:

Empaqueta una aplicación con sus dependencias: Los contenedores comparten el núcleo del sistema operativo anfitrión y sólo necesitan bibliotecas y archivos específicos para ejecutarse.

Contenedores Docker:

Ligeros y portátiles: Los contenedores son mucho más pequeños y rápidos de arrancar que las máquinas virtuales, lo que los hace ideales para microservicios y aplicaciones escalables.

Entornos coherentes: Los contenedores estandarizados garantizan un comportamiento coherente de las aplicaciones en diferentes máquinas.

Uso eficiente de los recursos: Compartir el kernel permite ejecutar muchos más contenedores en una sola máquina en comparación con las VM.

Contenedores Docker:

Contras:

Aislamiento limitado: Aunque están aislados, los contenedores comparten el kernel, lo que introduce algunos riesgos potenciales de seguridad en comparación con las máquinas virtuales.

Elección de SO limitada: Los contenedores suelen utilizar el sistema operativo del host, lo que limita su flexibilidad en comparación con las máquinas virtuales.

Utilizar máquinas virtuales para:

- Aplicaciones con dependencias conflictivas o estrictos requisitos de seguridad.
- Ejecución de varios sistemas operativos en una sola máquina.
- Cuando el aislamiento completo es crítico.

Utiliza contenedores Docker para:

- Microservicios y aplicaciones escalables.
- Despliegue de aplicaciones en entornos consistentes.
- Utilización eficiente de los recursos y despliegues rápidos.



¿Por qué es importante Docker en la ciencia de datos?



¿Por qué es importante Docker en la ciencia de datos?

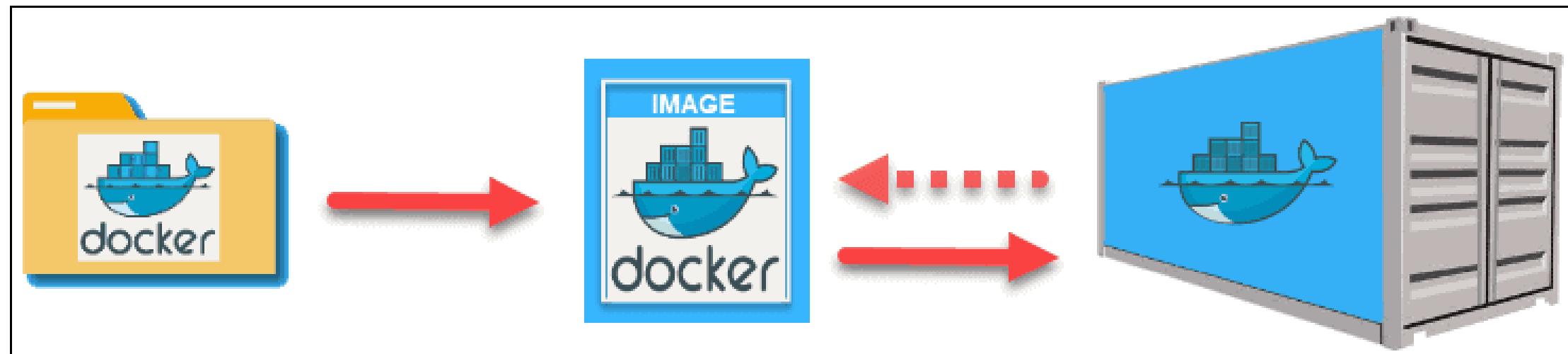
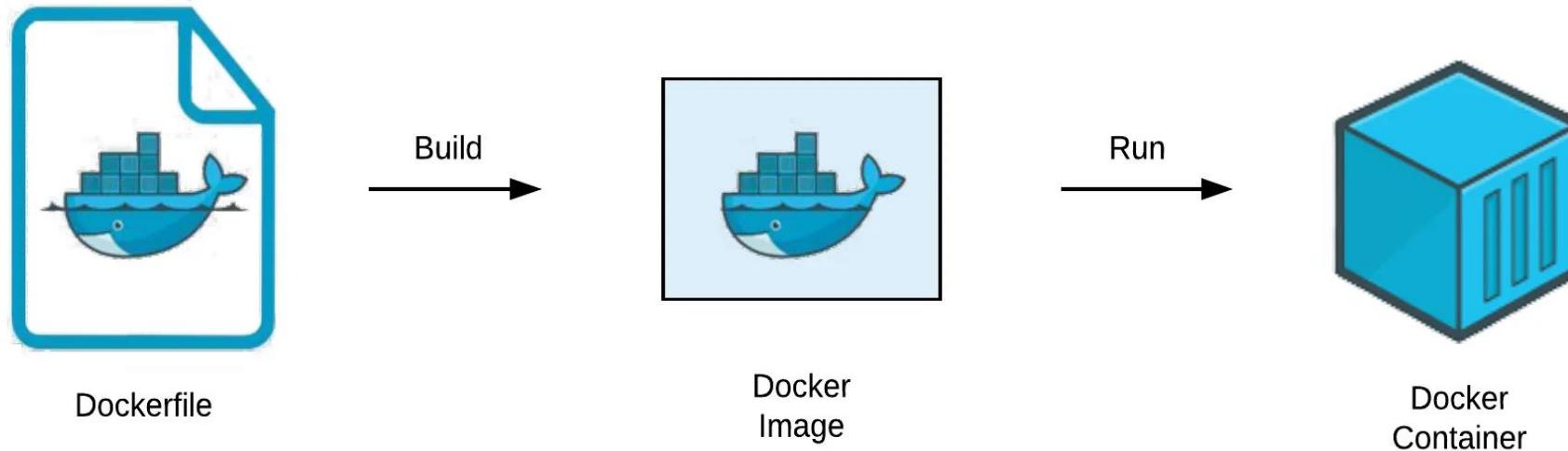
Suele pasar con mucha frecuencia que has desarrollado un modelo de aprendizaje automático y luego, cuando has cambiado de portátil, tu código se está rompiendo y te encuentras con una declaración infinita de "Error de importación" o "Módulo no encontrado". A veces, podría ser un error de versión debido a que se está tratando de ejecutar el código en una versión diferente de Python. La solución para esto es Docker.

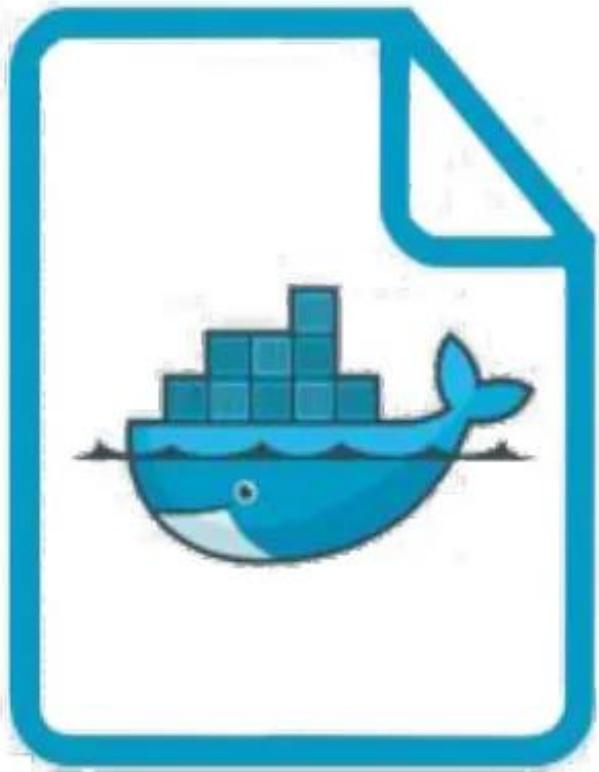
¿Por qué es importante Docker en la ciencia de datos?

Docker es una herramienta para crear y desplegar entornos aislados para ejecutar aplicaciones con sus dependencias. Básicamente, Docker facilita la escritura y ejecución de códigos sin problemas en otras máquinas con sistemas operativos diferentes, reuniendo el código y todas sus dependencias en un contenedor.

Este contenedor hace que el código sea autónomo e independiente del sistema operativo.

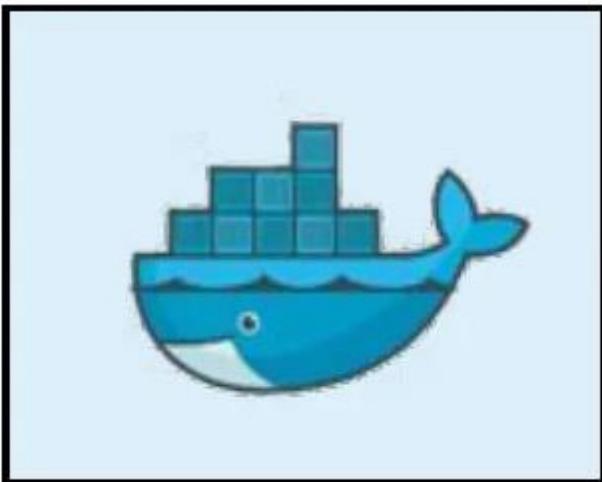
Terminología de Docker





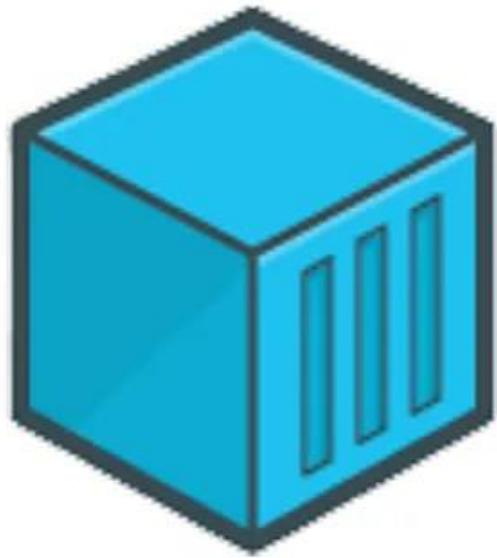
Dockerfile

Dockerfile - Un Dockerfile contiene todo el código para configurar un contenedor docker desde la descarga de la imagen docker hasta la configuración del entorno. Puedes pensar en él como si describiera la instalación completa del sistema operativo del sistema que quieras ejecutar.



Docker
Image

La imagen Docker es una plantilla de sólo lectura que contiene un conjunto de instrucciones para crear un contenedor que pueda ejecutarse en la plataforma Docker.



Docker
Container

Contenedor Docker - Un contenedor es una instancia ejecutable de una imagen. Puede crear, iniciar, detener, mover o eliminar un contenedor mediante la API o la CLI de Docker.

¿Por qué es importante Docker en la ciencia de datos?

Ejecutar modelos ML en contenedores Docker aporta una serie de ventajas:

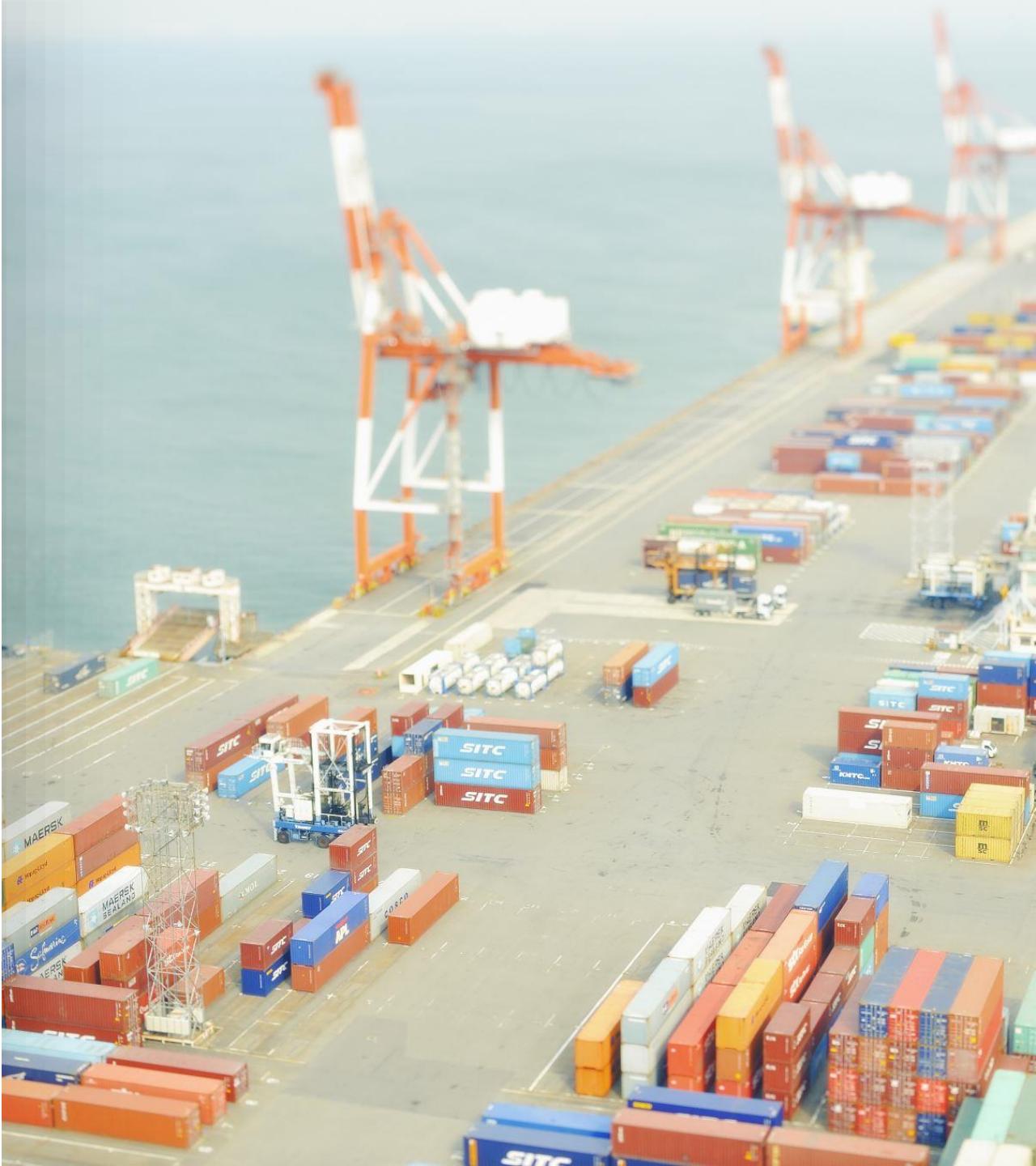
Reproducibilidad: Una vez que hayas probado tu aplicación en contenedores, puedes desplegarla en cualquier otro sistema en el que se ejecute Docker y puedes estar seguro de que tu aplicación funcionará exactamente igual que cuando la probaste.



¿Por qué es importante Docker en la ciencia de datos?

Agilidad: La portabilidad y las ventajas de rendimiento que ofrecen los contenedores pueden ayudarle a que su proceso de desarrollo sea más ágil y receptivo. Mejora sus procesos de integración continua y entrega continua.

Portabilidad: Esto significa que pasar del desarrollo local a un clúster de supercomputación es fácil. Puede evitar problemas de configuración.



Importancia de Docker en Big Data

Consistencia en los entornos: Permite crear entornos de ejecución consistentes que se comportan de la misma manera, independientemente del sistema operativo o la infraestructura subyacente.

Los contenedores Docker encapsulan las dependencias, librerías y configuraciones necesarias para que las aplicaciones de Big Data funcionen correctamente. Esto asegura que el código funcione igual en desarrollo, prueba y producción.

Importancia de Docker en Big Data

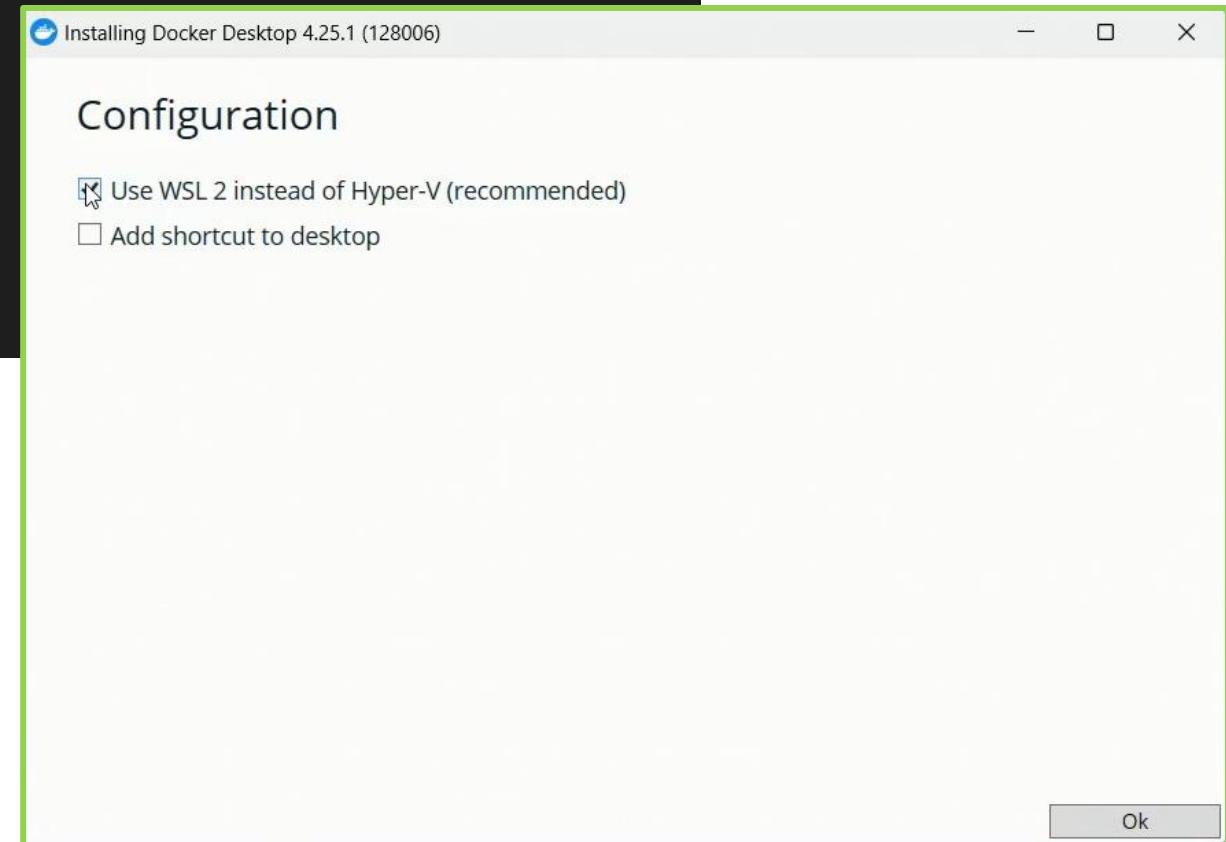
Docker facilita la escalabilidad al permitir que los nodos de procesamiento se creen y gestionen rápidamente en contenedores que pueden desplegarse en cualquier servidor o clúster.

Con Docker Swarm o Kubernetes, la orquestación de múltiples contenedores se simplifica, permitiendo escalar horizontalmente aplicaciones como Hadoop o Spark para adaptarse a cargas de trabajo cambiantes.

Verificar versión WSL en VS Code

PROBLEMS OUTPUT TERMINAL PORTS GITLENS AZURE SQL CONSOLE COMMENTS DEBUG CONSOLE

- PS C:\Users\juanj\OneDrive\Escritorio\Big-Data-2024> `wsl --version`
Versión de WSL: 2.3.24.0
Versión de kernel: 5.15.153.1-2
Versión de WSLg: 1.0.65
Versión de MSRDC: 1.2.5620
Versión de Direct3D: 1.611.1-81528511
Versión DXCore: 10.0.26100.1-240331-1435.ge-release
Versión de Windows: 10.0.22631.4317
- PS C:\Users\juanj\OneDrive\Escritorio\Big-Data-2024>



Instalar extensión en VS Code

The screenshot shows the Visual Studio Code interface with the Extensions sidebar open. The search bar at the top contains the text "Big-Data-2024". In the Extensions sidebar, the "Docker" extension by Microsoft is highlighted with a red box. The main view displays the "Docker" extension details page. The extension icon features a blue whale-like ship with a stack of squares above it. The title is "Docker v1.29.3" by Microsoft ([microsoft.com](#)). It has 38,395,485 installs and a 5-star rating from 95 reviews. A description below states: "Makes it easy to create, manage, and debug containerized applications." Below the description are buttons for "Disable" and "Uninstall" and a checked "Auto Update" option. At the bottom of the page are tabs for "DETAILS", "FEATURES", "CHANGELOG", and "DEPENDENCIES". The "Contributing" section is visible, along with a note about the contribution guidelines and a "Code of Conduct". The bottom navigation bar includes links for PROBLEMS, OUTPUT, TERMINAL (which is underlined), PORTS, GITLENS, AZURE, SQL CONSOLE, COMMENTS, and DEBUG CONSOLE.

Instalación de Docker en Windows

[Home](#) / [Manuals](#) / [Docker Desktop](#) / [Install](#) / [Windows](#)

Install Docker Desktop on Windows

Docker Desktop terms

Commercial use of Docker Desktop in larger enterprises (more than 250 employees OR more than \$10 million USD in annual revenue) requires a [paid subscription](#).

This page contains the download URL, information about system requirements, and instructions on how to install Docker Desktop for Windows.

[Docker Desktop for Windows - x86_64](#)

[Docker Desktop for Windows - Arm \(Beta\)](#)

For checksums, see [Release notes](#)

UI de Docker cuando se ejecuta desde Windows

Docker desktop PERSONAL

Containers [Give feedback](#)

Container CPU usage ⓘ Container memory usage ⓘ Show charts

No containers are running.

Search Only show running containers

<input type="checkbox"/>	Name	Image	Status	Port(s)	CPU (%)	Last started	Actions
<input type="checkbox"/>	proxy dcf993e5b110	nginx:<none>	Exited	8080:80	N/A	2 months ago	> ⋮ trash

Docker desktop PERSONAL

Containers [Give feedback](#)

Container CPU usage ⓘ Container memory usage ⓘ Show charts

No containers are running.

Search Only show running containers

<input type="checkbox"/>	Name	Image	Status	Port(s)	CPU (%)	Last started	Actions
<input type="checkbox"/>	proxy dcf993e5b110	nginx:<none>	Exited	8080:80	N/A	2 months ago	> ⋮ trash

Learning Center

Containers, volumes, extensions and more... **Ctrl+K**

Learning center

Walkthroughs

Quick hands-on guides to show you around

What is a container?
5 mins

How do I run a container?
6 mins

[View all](#)

AI/ML guides

Get started with AI/ML using Docker

GenAI Stack
Start your GenAI application using Neo4j, Langchain, Ollama, Python, and Docker Compose

Docker for beginners by language

45 min guides written for different programming languages

NodeJS

Python

Go

Java

C# (.NET)

Rust

[Request a guide ↗](#)

Docker Images

Images [Give feedback](#)

Local Hub

265.74 MB / 265.74 MB in use 2 images

Search Filter Sort

Name	Tag
nginx 5ef79149e0ec	latest
ubuntu edbfe74c41f8	latest

Images [Give feedback](#)

Local Hub

juanjorb23

Search

Tags

 juanjorb23/sitioweb	latest
-----------------------------------------------------------------------------------------------------------	--------

Volumes en Docker:

Volumes Give feedback						
<input type="checkbox"/>	Search	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	Name	Status	Created	Size	Scheduled exports <small>BETA</small>	Actions
<input type="checkbox"/>	03fee6fec6a1fe1efefb3136c54d8a6dde3d4a618cf4458329ffc77b971b1115	-	2 months ago	0 Bytes	Inactive	<input type="button"/> <input type="button"/>
<input type="checkbox"/>	20ef28bd50e42339859896a248ee28e04b4b7b1318101a4164f08767a1e6effd	-	2 months ago	300.4 MB	Inactive	<input type="button"/> <input type="button"/>
<input type="checkbox"/>	4806a44ba1e1c549d556d99acd8fe80ea9b82804a997834ced0c8c3335fa84b8	-	2 months ago	300.2 MB	Inactive	<input type="button"/> <input type="button"/>

Es un mecanismo para persistir datos generados o utilizados por contenedores. Los volúmenes se utilizan cuando los datos necesitan sobrevivir al ciclo de vida de un contenedor (es decir, cuando un contenedor se detiene o es eliminado, los datos persisten).

Settings - Configuración

The screenshot shows the Docker Desktop settings interface. The title bar includes the Docker Desktop logo, the word "PERSONAL", a search bar, and various icons. The main area has a sidebar with tabs: General (selected), Resources, Docker Engine, Builders, Kubernetes, Software updates, Extensions, Features in development, and Notifications. The "General" tab contains the following configuration options:

- Start Docker Desktop when you sign in to your computer
- Open Docker Dashboard when Docker Desktop starts
- Choose theme for Docker Desktop:
 Light Dark Use system settings
- Choose container terminal:
 Integrated System default
Determines which terminal is launched when opening the terminal from a container.
- Enable Docker terminal
- Enable Docker Debug by default [Learn more](#)
Active subscription required. [Upgrade](#)
- Expose daemon on tcp://localhost:2375 without TLS
Exposing daemon on TCP without TLS helps legacy clients connect to the daemon. It also makes yourself vulnerable to remote code execution attacks. Use with caution.
- Use the WSL 2 based engine (Windows Home can only run the WSL 2 backend)
WSL 2 provides better performance than the Hyper-V backend. [Learn more](#)
- Add the *.docker.internal names to the host's /etc/hosts file (Requires password)
Lets you resolve *.docker.internal DNS names from both the host and your containers. [Learn more](#)
- Use containerd for pulling and storing images [Give feedback](#)
The containerd image store enables native support for multi-platform images, attestations, Wasm, and more.
- Send usage statistics

Docker CLI

Desde vs code abrimos una terminal de wsl y ejecutamos: **docker --version**

PROBLEMS OUTPUT TERMINAL PORTS GITLENS AZURE SQL CONSOLE COMMENTS DEBUG CONSOLE

```
balrodjuan@balrodjj:/mnt/c/Users/juanj/OneDrive/Escritorio/Big-Data-2024$ docker --version
Docker version 27.2.0, build 3ab4256
balrodjuan@balrodjj:/mnt/c/Users/juanj/OneDrive/Escritorio/Big-Data-2024$ █
```

Para acceder a toda la documentación (todos los comandos) se escribe: **docker**

```
balrodjuan@balrodjj:/mnt/c/Users/juanj/OneDrive/Escritorio/Big-Data-2024$ docker
Usage: docker [OPTIONS] COMMAND
A self-sufficient runtime for containers
Common Commands:
```

El comando docker info: proporciona un resumen detallado de la configuración actual de Docker, incluyendo información sobre el estado del servidor, los recursos disponibles y la configuración del entorno de Docker.

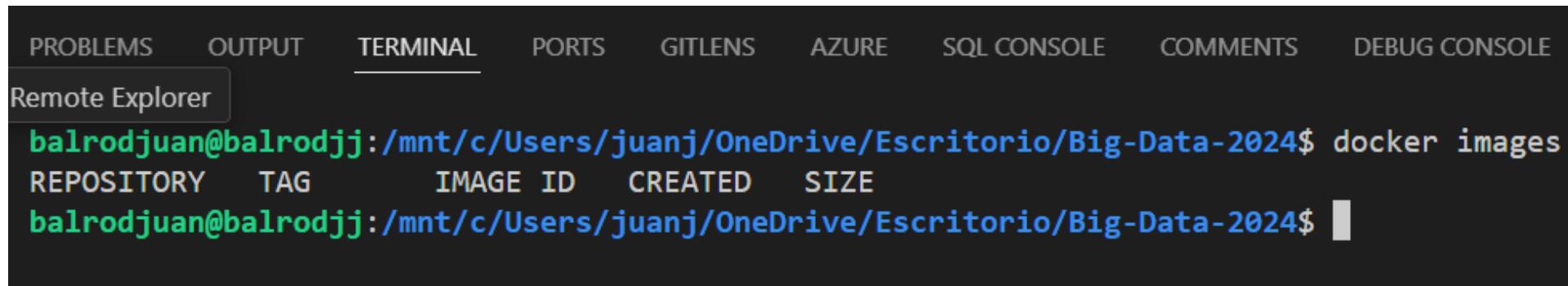
```
balrodjuan@balrodjj:/mnt/c/Users/juanj/OneDrive/Escritorio/Big-Data-2024$ docker info
Client:
  Version: 27.2.0
  Context: default
  Debug Mode: false
  Plugins:
    buildx: Docker Buildx (Docker Inc.)
      Version: v0.16.2-desktop.1
      Path: /usr/local/lib/docker/cli-plugins/docker-buildx
    compose: Docker Compose (Docker Inc.)
      Version: v2.29.2-desktop.2
      Path: /usr/local/lib/docker/cli-plugins/docker-compose
    debug: Get a shell into any image or container (Docker Inc.)
      Version: 0.0.34
      Path: /usr/local/lib/docker/cli-plugins/docker-debug
    desktop: Docker Desktop commands (Alpha) (Docker Inc.)
      Version: v0.0.15
      Path: /usr/local/lib/docker/cli-plugins/docker-desktop
    dev: Docker Dev Environments (Docker Inc.)
      Version: v0.1.2
      Path: /usr/local/lib/docker/cli-plugins/docker-dev
    extension: Manages Docker extensions (Docker Inc.)
      Version: v0.2.25
      Path: /usr/local/lib/docker/cli-plugins/docker-extension
    feedback: Provide feedback, right in your terminal! (Docker Inc.)
```

Actividad:

Escribe docker info y luego explica lo que contiene la salida. Investigar para que se usa. Guarda tus resultados en tus apuntes

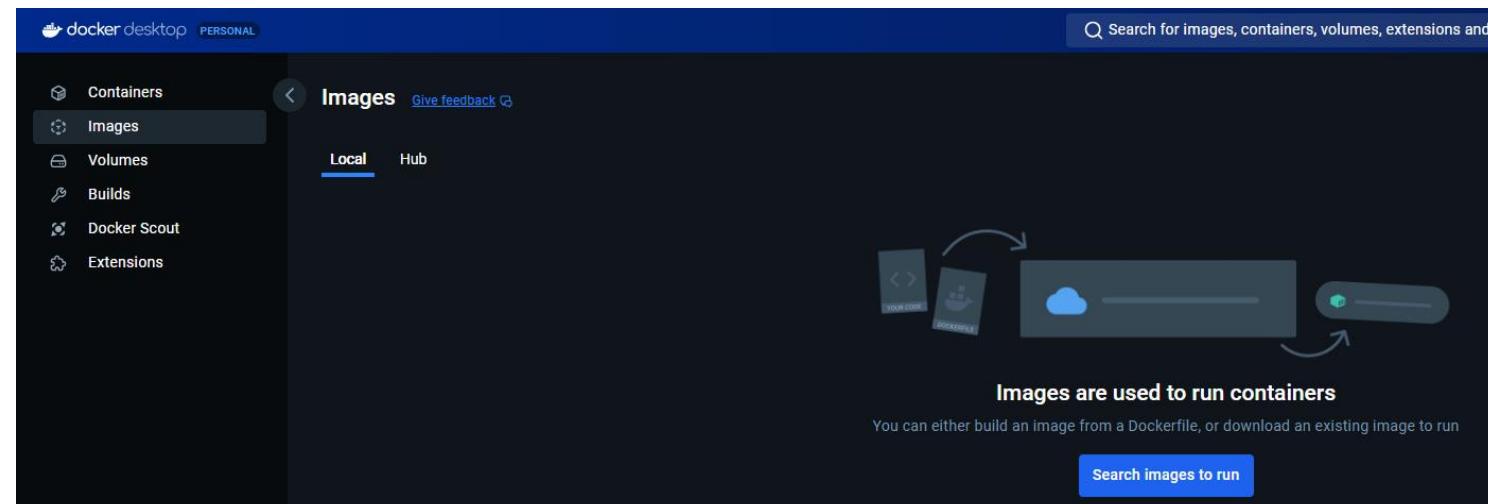
docker images

Se usa para listar las imágenes disponibles localmente en tu sistema Docker. Proporciona un resumen de todas las imágenes que están almacenadas en tu máquina, mostrando detalles clave como el nombre de la imagen, la etiqueta, el ID, la fecha de creación y el tamaño.



```
PROBLEMS OUTPUT TERMINAL PORTS GITLENS AZURE SQL CONSOLE COMMENTS DEBUG CONSOLE
Remote Explorer
balrodjuan@balrodjj:/mnt/c/Users/juanj/OneDrive/Escritorio/Big-Data-2024$ docker images
REPOSITORY      TAG          IMAGE ID   CREATED     SIZE
balrodjuan@balrodjj:/mnt/c/Users/juanj/OneDrive/Escritorio/Big-Data-2024$
```

Desde Docker Desktop



Documentación de docker images: docker images --help

```
balrodjuan@balrodjj:/mnt/c/Users/juanj/OneDrive/Escritorio/Big-Data-2024$ docker images --help
```

```
Usage: docker images [OPTIONS] [REPOSITORY[:TAG]]
```

```
List images
```

```
Aliases:
```

```
  docker image ls, docker image list, docker images
```

```
Options:
```

-a, --all	Show all images (default hides intermediate images)
--digests	Show digests
-f, --filter filter	Filter output based on conditions provided
--format string	Format output using a custom template: 'table': Print output in table format with column headers (default) 'table TEMPLATE': Print output in table format using the given Go template 'json': Print in JSON format 'TEMPLATE': Print output using the given Go template. Refer to https://docs.docker.com/go/formatting/ for more information about formatting output with templates
--no-trunc	Don't truncate output
-q, --quiet	Only show image IDs
--tree	List multi-platform images as a tree (EXPERIMENTAL)

docker run

Permite crear y ejecutar contenedores a partir de una imagen. Al ejecutar este comando, Docker hace lo siguiente:

Descargar la imagen (si no está disponible localmente).

Crear un contenedor basado en esa imagen.

Ejecutar un comando dentro del contenedor.

Asignar recursos (como redes, volúmenes, etc.) según las opciones proporcionadas.

Sintaxis básica:

```
docker run [opciones] imagen [comando]
```

Ejemplo, vamos a ejecutar Ubuntu desde una imagen oficial de Docker Hub

```
balrodjuan@balrodjj:/mnt/c/Users/juanj/OneDrive/Escritorio/Big-Data-2024$ docker run -it ubuntu /bin/bash
```

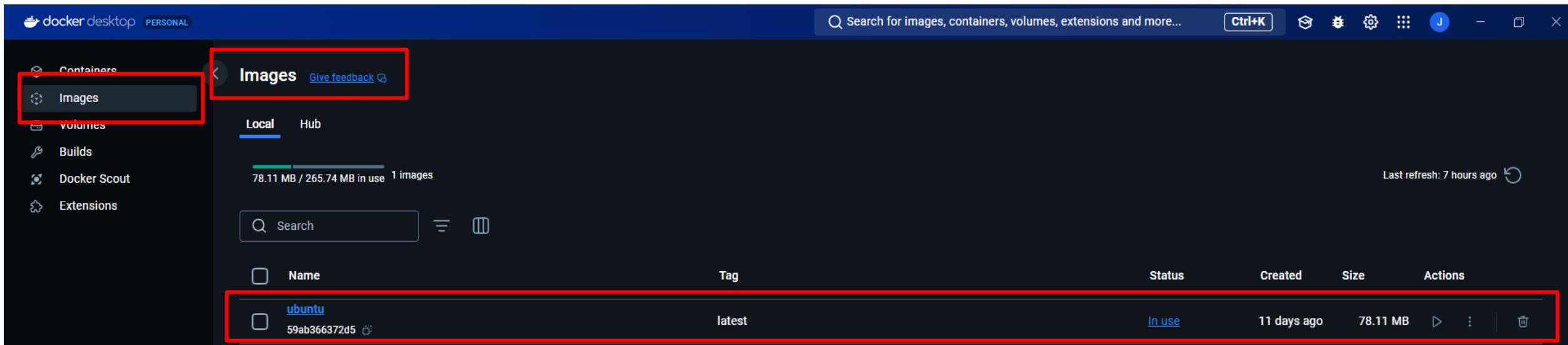
```
balrodjuan@balrodjj:/mnt/c/Users/juanj/OneDrive/Escritorio/Big-Data-2024$ docker run -it ubuntu /bin/bash
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
ff65ddf9395b: Pull complete
Digest: sha256:99c35190e22d294cdace2783ac55effc69d32896daaa265f0bbbedbcde4fbe3e5
Status: Downloaded newer image for ubuntu:latest
root@1a428a62008c:/#
```

```
root@1a428a62008c:/# ls
bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@1a428a62008c:/#
```

```
docker run -it ubuntu /bin/bash
```

- -it: Ejecuta el contenedor en modo interactivo y con una terminal.
- ubuntu: Nombre de la imagen que estás utilizando.
- ./bin/bash: El comando que se ejecuta dentro del contenedor, en este caso, abre una shell de bash.

En Docker Desktop, en ‘Images’ se ve lo siguiente:



The screenshot shows the Docker Desktop interface with the 'Images' tab selected. A red box highlights the 'Images' tab in the sidebar and the 'ubuntu' image entry in the main table.

Docker Desktop PERSONAL

Search for images, containers, volumes, extensions and more... Ctrl+K

Containers Images Give feedback

Local Hub

78.11 MB / 265.74 MB in use 1 images Last refresh: 7 hours ago

Builds Docker Scout Extensions

Search

Name	Tag	Status	Created	Size	Actions
ubuntu	latest	In use	11 days ago	78.11 MB	⋮
59ab366372d5					trash

Mientras que, en ‘Containers’ se ve lo siguiente:

The screenshot shows the Docker Desktop application interface. The left sidebar has options: Containers (selected and highlighted with a red box), Images, Volumes, Builds, Docker Scout, and Extensions. The main area is titled 'Containers' with a 'Give feedback' link. It displays 'Container CPU usage' and 'Container memory usage', both stating 'No containers are running.' There is a 'Show charts' link. A search bar and a 'Only show running containers' toggle are present. The table below lists one container:

<input type="checkbox"/>	Name	Image	Status	Port(s)	CPU (%)	Last started	Actions
<input type="checkbox"/>	exciting_ishizaka 1a428a62008c	ubuntu:<none>	Exited		N/A	1 hour ago	

Ejecutar un contenedor y eliminarlo al salir

```
docker run --rm ubuntu echo "Hola desde Docker"
```

Ejecuta un contenedor temporal que se elimina automáticamente cuando termina su ejecución.

- `--rm`: Elimina automáticamente el contenedor cuando termina.
- `echo "Hola desde Docker"`: El comando que se ejecuta dentro del contenedor, que imprime un mensaje.

```
balrodjuan@balrodjj:/mnt/c/Users/juanj/OneDrive/Escritorio/Big-Data-2024$ docker run --rm ubuntu echo "Hola desde Docker"
Hola desde Docker
balrodjuan@balrodjj:/mnt/c/Users/juanj/OneDrive/Escritorio/Big-Data-2024$
```

Si no se elimina con el comando anterior hacer lo siguiente:

```
PROBLEMS    OUTPUT    TERMINAL    PORTS    GITLENS    AZURE    SQL CONSOLE    COMMENTS    DEBUG CONSOLE

balrodjuan@balrodjj:/mnt/c/Users/juanj/OneDrive/Escritorio/Big-Data-2024$ docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS          NAMES
1a428a62008c        ubuntu              "/bin/bash"         3 hours ago       Exited (0) 2 hours ago
balrodjuan@balrodjj:/mnt/c/Users/juanj/OneDrive/Escritorio/Big-Data-2024$ docker stop 1a428a62008c
1a428a62008c
balrodjuan@balrodjj:/mnt/c/Users/juanj/OneDrive/Escritorio/Big-Data-2024$ docker rm 1a428a62008c
1a428a62008c
balrodjuan@balrodjj:/mnt/c/Users/juanj/OneDrive/Escritorio/Big-Data-2024$ docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS          NAMES
balrodjuan@balrodjj:/mnt/c/Users/juanj/OneDrive/Escritorio/Big-Data-2024$
```

Si nos vamos a Docker Desktop podemos verificar que se borró el ‘container’

The screenshot shows the Docker Desktop application window. The top navigation bar includes the Docker logo, 'docker desktop' text, a 'PERSONAL' badge, a search bar with placeholder text 'Search for images, containers, volumes, extensions and more...', and keyboard shortcut 'Ctrl+K'. On the left, a sidebar menu lists 'Containers' (selected), 'Images', 'Volumes', 'Builds', 'Docker Scout', and 'Extensions'. The main content area is titled 'Containers' with a 'Give feedback' link. It features a central graphic of three blue cylinders representing containers, with a small green cube icon in the middle cylinder. Below the graphic, the text 'Your running containers show up here' is displayed, followed by the definition 'A container is an isolated environment for your code'. At the bottom, there are two cards: 'What is a container?' (5 mins) with a question mark icon, and 'How do I run a container?' (6 mins) with a code snippet icon.

docker desktop PERSONAL

Containers Images Volumes Builds Docker Scout Extensions

Containers Give feedback

Search for images, containers, volumes, extensions and more... Ctrl+K

Your running containers show up here

A container is an isolated environment for your code

What is a container? 5 mins

How do I run a container? 6 mins

View more in the Learning center

En el sitio web oficial de Docker se pueden descargar las imágenes

The screenshot shows the Docker Hub homepage with a dark blue header. The header includes the Docker Hub logo, navigation links for Explore, Repositories, Organizations, and Usage, and various user interface icons. A banner at the top reads "New More Docker. Easy Access. New Streamlined Plans. Learn more." Below the header is a search bar with the placeholder "Search Docker Hub" and a keyboard shortcut "ctrl+K". The main content area features several sections:

- Trusted content**: Includes links to Docker Official Image, Verified Publisher, and Sponsored OSS.
- Categories**: Lists API Management, Content Management System, Data Science, Databases & Storage, Languages & Frameworks, Integration & Delivery, Internet of Things, Machine Learning & AI, Message Queues, Monitoring & Observability, Networking, Operating Systems, Security, Web Servers, and Developer Tools.
- Spotlight**: Three cards highlighting Docker Build Cloud (Cloud Development), Docker and Hugging Face (AI/ML Development), and Docker Scout (Software Supply Chain).
- Machine Learning & AI**: Four cards for tensorflow/tensorflow, pytorch/pytorch, langchain/langchain, and ollama/ollama.
- Trending this week**: A section showing trending repositories.

Ejemplo básico de creación de un contenedor básico para un portfolio o web personal.

1. Creamos una carpeta en local para nuestro porfolio

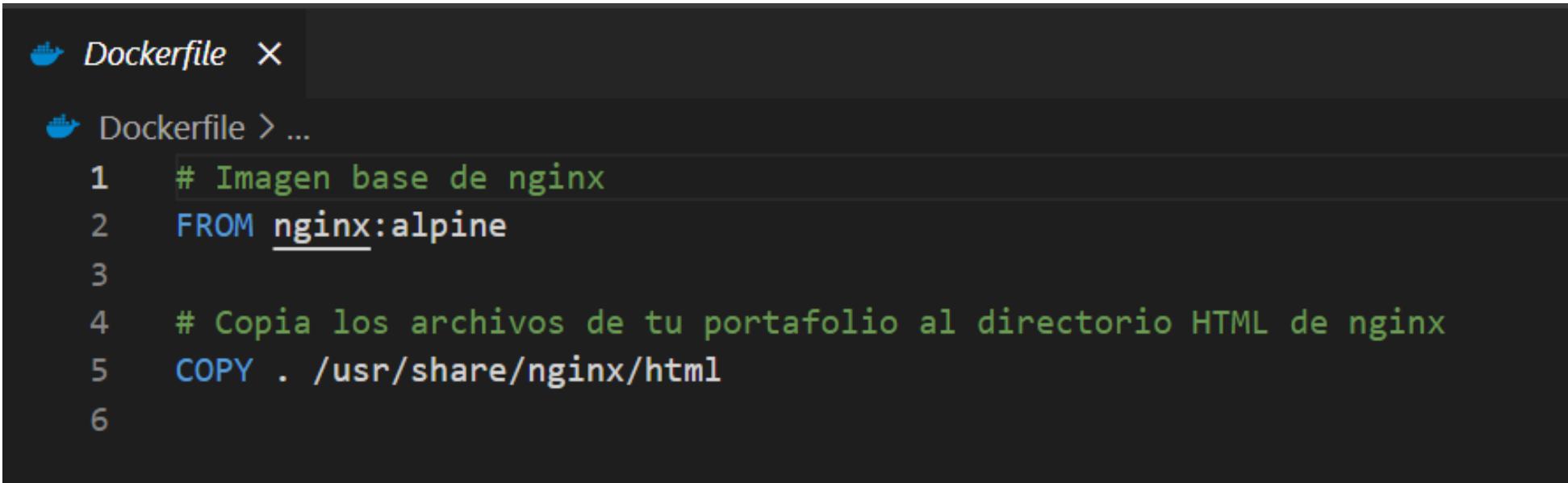
```
balrodjuan@balrodjj:/mnt/c/Users/juanj/OneDrive/Escritorio$ mkdir miport  
balrodjuan@balrodjj:/mnt/c/Users/juanj/OneDrive/Escritorio$ cd miport  
balrodjuan@balrodjj:/mnt/c/Users/juanj/OneDrive/Escritorio/miport$ █
```

2. Creamos un archivo **Dockerfile** dentro de la carpeta miport

```
balrodjuan@balrodjj:/mnt/c/Users/juanj/OneDrive/Escritorio/miport$ touch Dockerfile█
```

```
balrodjuan@balrodjj:/mnt/c/Users/juanj/OneDrive/Escritorio/miport$ code Dockerfile█
```

Contenido del Dockerfile



A screenshot of a code editor window titled "Dockerfile". The file contains the following Dockerfile code:

```
1 # Imagen base de nginx
2 FROM nginx:alpine
3
4 # Copia los archivos de tu portafolio al directorio HTML de nginx
5 COPY . /usr/share/nginx/html
6
```

3. Crear la plantilla html para el portfolio

3. Crear plantilla html para nuestro portfolio

3.1 Crear el archivo index.html en la misma carpeta del Dockerfile

```
balrodjuan@balrodjj:/mnt/c/Users/juanj/OneDrive/Escritorio/miport$ touch index.html  
balrodjuan@balrodjj:/mnt/c/Users/juanj/OneDrive/Escritorio/miport$ code index.html  
balrodjuan@balrodjj:/mnt/c/Users/juanj/OneDrive/Escritorio/miport$ █
```

El contenido del archivo index.html lo tienes en el bloc de notas

4. Construir la imagen de Docker

Desde la terminal y estando ubicados en el mismo directorio del Dockerfile escribimos:

docker build -t miport .

```
balrodjuan@balrodjj:/mnt/c/Users/juanj/OneDrive/Escritorio/miport$ docker build -t miport .
[+] Building 7.2s (8/8) FINISHED                                            docker:default
=> [internal] load build definition from Dockerfile                      0.1s
=> => transferring dockerfile: 176B                                         0.1s
=> [internal] load metadata for docker.io/library/nginx:alpine          3.5s
=> [auth] library/nginx:pull token for registry-1.docker.io              0.0s
=> [internal] load .dockerignore                                         0.0s
=> => transferring context: 2B                                           0.0s
=> [1/2] FROM docker.io/library/nginx@sha256:2140dad235c130ac861018a4e13a6bc8aea3a35f3a40e20c1b060d51a7efd250 3.2s
=> => resolve docker.io/library/nginx@sha256:2140dad235c130ac861018a4e13a6bc8aea3a35f3a40e20c1b060d51a7efd250 0.0s
=> => sha256:2140dad235c130ac861018a4e13a6bc8aea3a35f3a40e20c1b060d51a7efd250 9.07kB / 9.07kB 0.0s
=> => sha256:43c4264eed91be63b206e17d93e75256a6097070ce643c5e8f0379998b44f170 3.62MB / 3.62MB 0.9s
=> => sha256:596d53a7de8832c0963cd374bf19a0a1ca2284c80329e1a1462c4f51035ae0c8 629B / 629B 0.5s
=> => sha256:ae136e431e76e12e5d84979ea5e2ffff4dd9589c2435c8bb9e33e6c3960111d3 2.50kB / 2.50kB 0.0s
```

`docker build -t mi_portafolio .`

Cuando ejecutas este comando en el mismo directorio donde está el Dockerfile, Docker hará lo siguiente:

- Buscará el archivo Dockerfile en el directorio actual (representado por el .).
- Leerá las instrucciones en el Dockerfile.
- Descargará la imagen base (`nginx:alpine`) si no está disponible localmente.
- Ejecutará los comandos en el Dockerfile, como copiar archivos o instalar dependencias.
- Construirá una nueva imagen con los cambios realizados.
- Asignará el nombre y la etiqueta `miport:latest` a la nueva imagen.

Resultado

Después de ejecutar el comando, tendrás una nueva imagen de Docker llamada miport en tu sistema. Puedes verificar que se ha creado con:
docker images

```
balrodjuan@balrodjj:/mnt/c/Users/juanj/OneDrive/Escritorio/miport$ docker images
REPOSITORY      TAG          IMAGE ID       CREATED        SIZE
miport          latest        8d737369c9fd   8 minutes ago  47MB
ubuntu           latest        59ab366372d5   11 days ago   78.1MB
balrodjuan@balrodjj:/mnt/c/Users/juanj/OneDrive/Escritorio/miport$ █
```

5. Ejecutar el Contenedor Docker

5.1. Utilizando la imagen que has creado ejecuta el contenedor Docker.

```
balrodjuan@balrodjj:/mnt/c/Users/juanj/OneDrive/Escritorio/miport$ docker run -d -p 8080:80 miport  
22265504da35af47ab5bcd970931da5a60c09463a9823c19abaa74fde6fe60f2  
balrodjuan@balrodjj:/mnt/c/Users/juanj/OneDrive/Escritorio/miport$ █
```

5.2. Accede a tu portfolio en el navegador web.

<http://localhost:8080>

Vista del portafolio desde el navegador:

Bienvenido a Mi Portafolio

Proyecto 1

Descripción del proyecto 1.

Proyecto 2

Descripción del proyecto 2.

6. Publicar en GitHub.

6.1 Iniciar en local:

```
juanj@balrodjj MINGW64 ~/OneDrive/Escritorio/miport
● $ git init
Initialized empty Git repository in C:/Users/juanj/OneDrive/Escritorio/miport/.git/
juanj@balrodjj MINGW64 ~/OneDrive/Escritorio/miport (master)
● $ git add .
warning: in the working copy of 'Dockerfile', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'index.html', LF will be replaced by CRLF the next time Git touches it
juanj@balrodjj MINGW64 ~/OneDrive/Escritorio/miport (master)
● $ git commit -m "Inicial commit de mi portafolio"
[master (root-commit) 3e1bea5] Inicial commit de mi portafolio
 2 files changed, 36 insertions(+)
 create mode 100644 Dockerfile
 create mode 100644 index.html
```

6.2 Crear un nuevo repositorio en GitHub

The screenshot shows a GitHub repository page for a repository named "miport". The repository is public. At the top, there are buttons for "Pin" and "Unwatch" with a count of 1. Below the header, there are buttons for "main" (with a dropdown arrow), "1 Branch" (with a dropdown arrow), and "Tags". A search bar with a magnifying glass icon and the placeholder "Go to file" is followed by a "t" button and a "+" button. A green "Code" button with a dropdown arrow is also present. The main content area shows a single commit by "rodbalza" titled "Initial commit" with hash "51a9393 · now" and "1 Commit". Below the commit, there is a file listing for "README.md" with the status "Initial commit" and "now". At the bottom, there is a preview of the "README" file content, which contains the word "miport".

miport Public

Pin Unwatch 1

main ▾ 1 Branch Tags

Go to file t + Code ▾

rodbalza Initial commit 51a9393 · now 1 Commit

README.md Initial commit now

README

miport

6.2 Agrega el repositorio remoto

```
juanj@balrodjj MINGW64 ~/OneDrive/Escritorio/miport (master)
$ git remote add origin https://github.com/rodbalza/miport.git
```

6.3 Subir los archivos al repositorio remoto

```
juanj@balrodjj MINGW64 ~/OneDrive/Escritorio/miport (master)
● $ git push -u origin master
  Enumerating objects: 4, done.
  Counting objects: 100% (4/4), done.
  Delta compression using up to 8 threads
  Compressing objects: 100% (4/4), done.
  Writing objects: 100% (4/4), 766 bytes | 766.00 KiB/s, done.
  Total 4 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote:     https://github.com/rodbalza/miport/pull/new/master
remote:
To https://github.com/rodbalza/miport.git
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.
```

Practica .

Visita el sitio web de Docker Hub busca imágenes de algún video juego muy ligero que puedas configurar y ejecutar usando Docker run.











