

# Introducción a la IA con contenedores de alta eficiencia

[@nonodev96/taller-gdg](https://twitter.com/nonodev96/taller-gdg)

## Quien soy

Soy Antonio Mudarra Machuca investigador en la Universidad de Jaén en el grupo de investigación SIMIDAT.



# Objetivos del taller

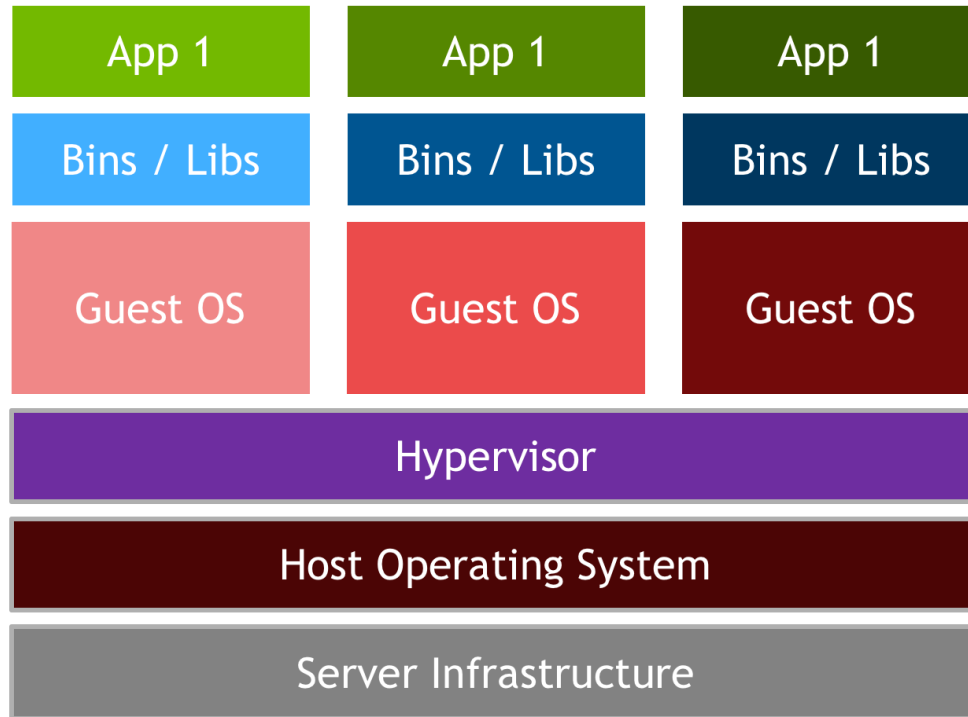
Mostrar las capacidades de **docker** para la ejecución de modelos de IA, simplificando todo el proceso de configuración de distintos entornos de desarrollo y ejecución.

- ✓ Introducción a docker y contenedores especializados en IA
- ✓ Comprender la diferencia entre imágenes y contenedores
- ✓ Configuración de entornos con PyTorch, TensorFlow y herramientas clave
- ✓ Conocer recursos de nvidia para el desarrollo y despliegue de modelos
- ✓ Ejecución de modelos LLM en tu propio ordenador con ollama
  - Breve introducción a la seguridad en modelos de IA
  - Entender y manejar docker, gestionar recursos de un contenedor
  - Conocer otras herramientas como ollama o traefik

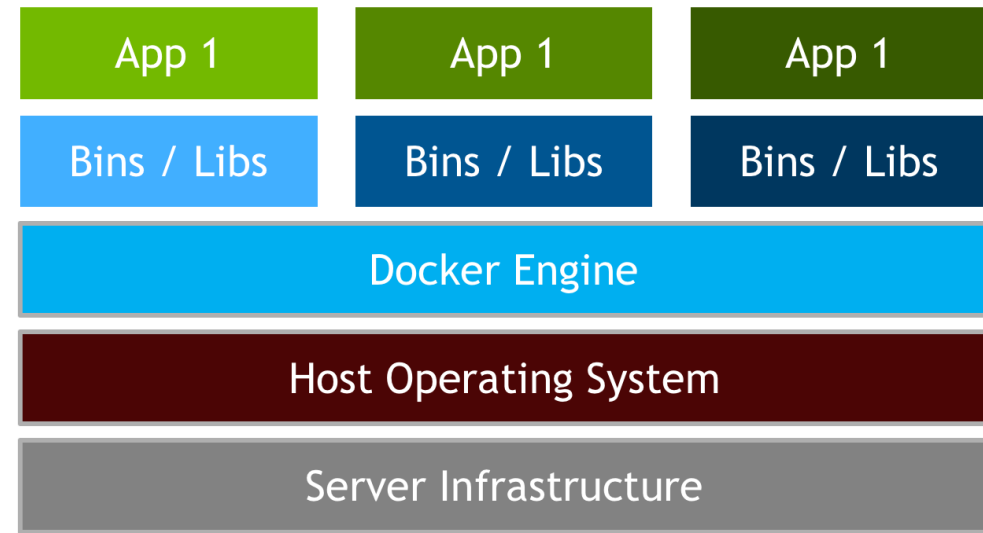
# Motivación

- Facilidad de despliegue
- Aislamiento de dispositivos individuales
- Ejecución en entornos heterogéneos de controladores/toolkits
- Sólo requiere la instalación del controlador NVIDIA en el host
- Facilita la colaboración: compilaciones reproducibles, rendimiento reproducible, resultados reproducibles

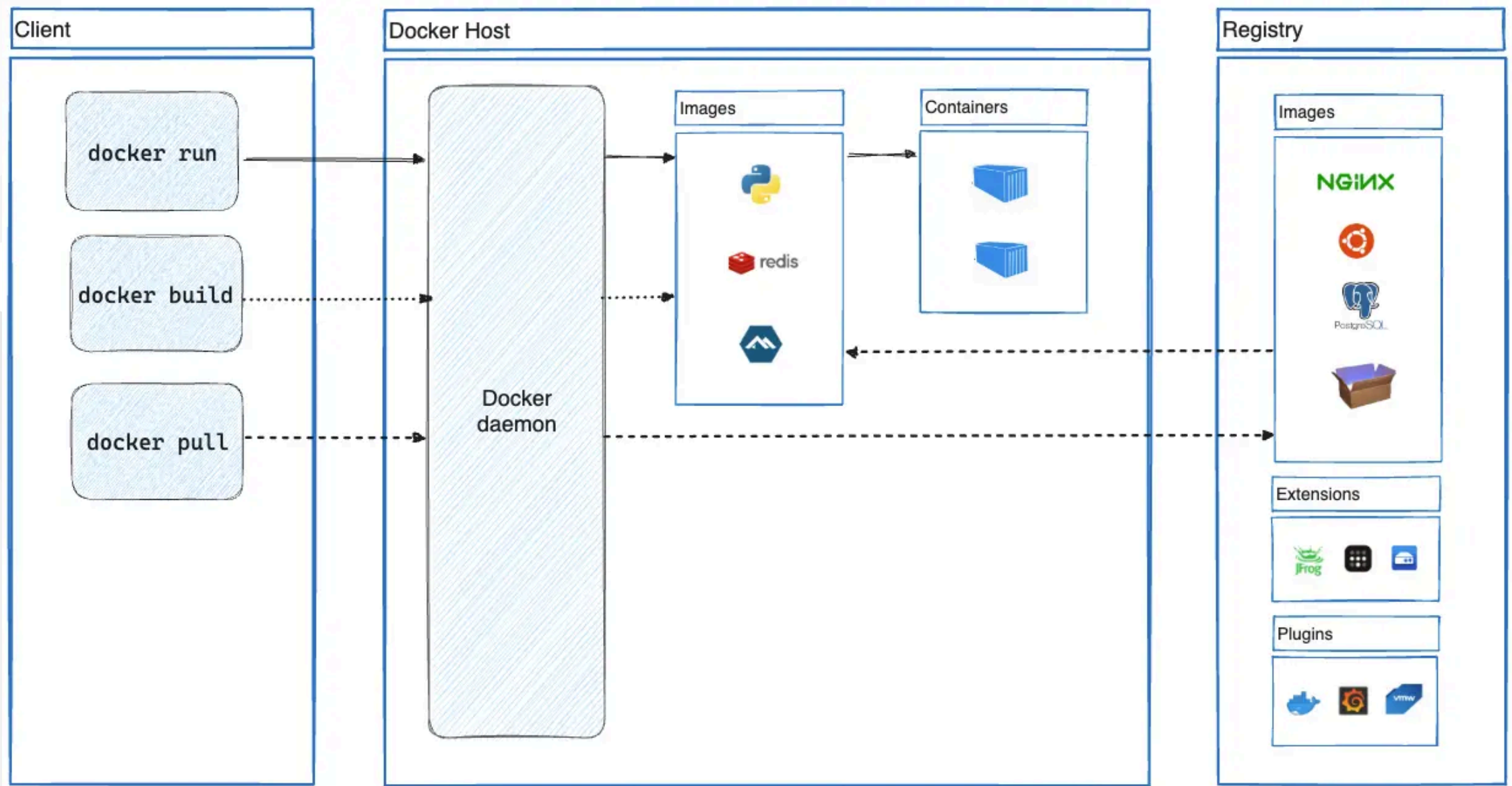
# Introducción a docker



**VIRTUAL MACHINES**



**CONTAINERS**



## Instalación de docker engine en Ubuntu

- [Install Docker Engine on Ubuntu](#)

## Descarga de los paquetes y actualización de las dependencias

```
sudo apt-get update
sudo apt-get install ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc

echo \
"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu \
$(. /etc/os-release && echo "${UBUNTU_CODENAME:-$VERSION_CODENAME}") stable" | \
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

sudo apt-get update
```



## Instalación de los paquetes de la comunidad en ubuntu

```
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
```

## Ejecución de una imagen para crear un contenedor

```
sudo docker run hello-world
```

Modificar los grupos para no necesitar permisos sudo al desplegar los contenedores.

```
sudo groupadd docker  
sudo usermod -aG docker $USER  
newgrp docker          # Activamos los cambios del grupo  
docker run hello-world # ya no hace falta ejecutar con permisos sudo
```

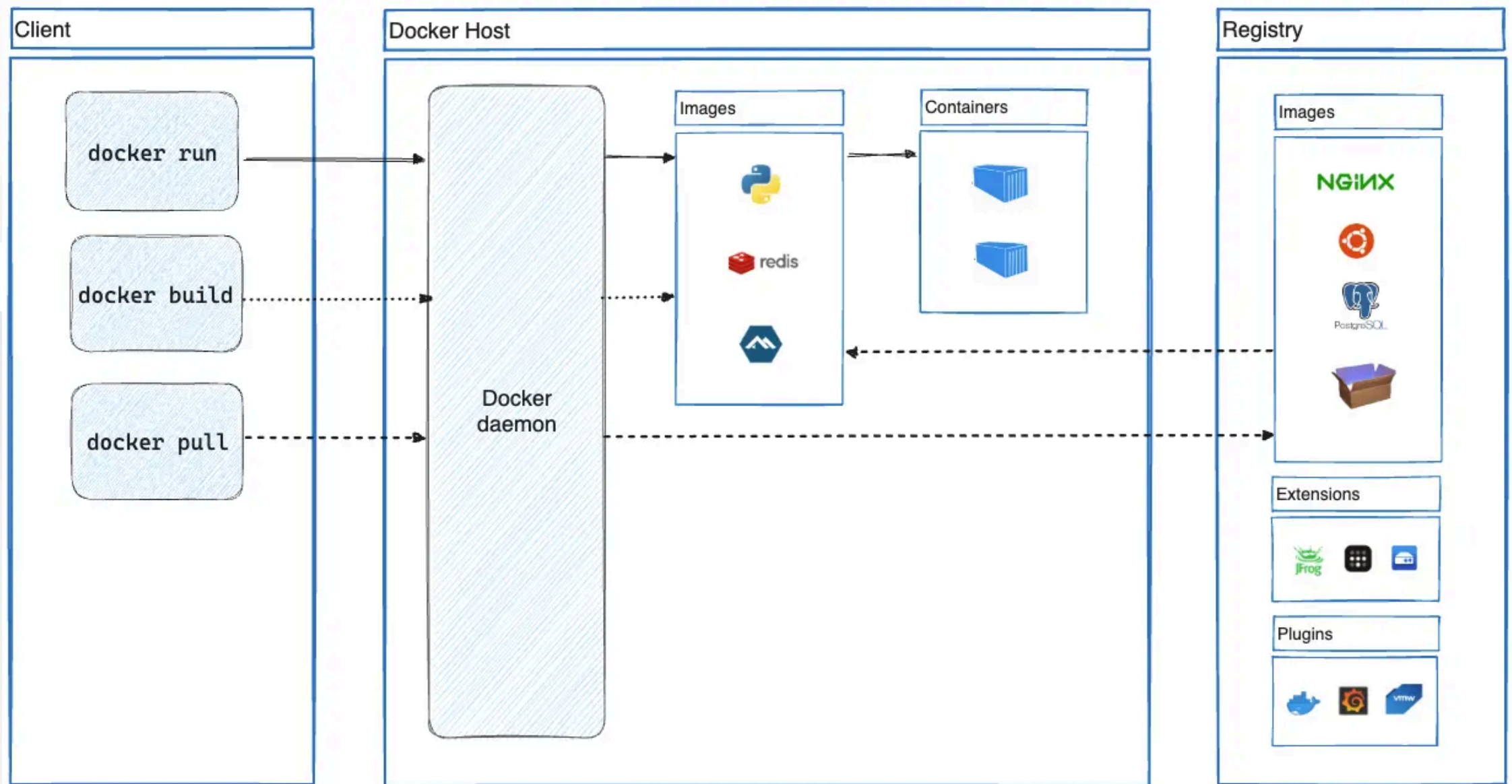
Si te aparece el siguiente warning es posible que se deba a que has ejecutado un contenedor previamente con permisos de administrador, puedes corregirlo modificando los permisos de la carpeta `~/.docker/`

```
WARNING: Error loading config file: /home/user/.docker/config.json -  
stat /home/user/.docker/config.json: permission denied
```

```
# eliminas o cambias los permisos  
sudo rm -rf /home/"$USER"/.docker  
  
sudo chown "$USER":"$USER" /home/"$USER"/.docker -R  
sudo chmod g+rx "/home/$HOME/.docker" -R
```

Instalación de docker desktop en windows, siguiente, siguiente, siguiente.

- [Install Docker Desktop on Windows](#)



# Creación de imágenes con docker

Fichero `Dockerfile` genérico.

```
# Definición de la imagen de docker de la que partir
FROM ubuntu:22.04

# Espacio de trabajo donde se iniciará el contenedor una vez creada la imagen
WORKDIR /workspace_nonodev96

# Ejemplo de comando para la creación de la imagen
RUN sudo apt update
RUN sudo apt install screen -y

# Ejemplo de ENTRYPOINT con CMD
ENTRYPOINT ["/bin/echo"] # Por defecto ENTRYPOINT es `/bin/sh -c`
CMD ["hello world!"]
```

# NVIDIA CUDA y Librerías

Instalación de CUDA, accedemos a la web para estudiar como instalar el kit de desarrollo de CUDA de nvidia, podemos descargar los drivers desde [CUDA Toolkit 12.8](#)

Para ver todo el listado de productos de nvidia con soporte de CUDA podemos acceder a la web [cuda-gpus](#).

- [CUDA GUIDE WINDOWS](#)
- [CUDA GUIDE LINUX](#)

## Sistemas operativos

Para Windows podemos abrir el panel de dispositivos con `control /name Microsoft.DeviceManager` > `Adaptadores de pantalla`.

Con Debian y derivados podemos ver la gráfica con `lspci | grep VGA` o con el paquete `hwinfo` instalando (`sudo apt install hwinfo`) y comprobando el hardware (`sudo hwinfo --gfxcard`).

```
lspci | grep VGA
lspci | grep -i nvidia
sudo apt install hwinfo
sudo hwinfo --gfxcard
```


Para ambos casos windows o linux, se instala la orden `nvidia-smi` (NVIDIA System Management Interface).

Este nos permite ver que hardware tiene nuestro equipo y como lo está usando. En la parte superior nos indica la versión máxima soportada por los drivers de nuestra tarjeta, no indica la versión instalada.

```
nvidia-smi
nvcc --version # Este es el compilador, no viene con los drivers
```



## Sistemas operativos compatibles con el toolkit de CUDA:

- Ubuntu 20.04, 22.04 y 24.04  Recomendado
- Microsoft Windows 11 24H2, 22H2-SV2 y 23H2
- Microsoft Windows 10 22H2
- Microsoft Windows WSL 2
- Debian 11 y 12
- RHEL / Rocky, KylinOS, Fedora, SLES, OpenSUSE, Amazon Linux, Azure Linux CM2.

## Paquetes que incluye

⚠️ Compatibilidad con los distintos sistemas operativos [cuda-toolkit-release-notes](#)

- CUDA
  - CUDA Driver
  - CUDA Runtime (cudart)
  - CUDA Math Library (math.h)
  - etc
- cuDNN
  - CUDA Deep Neural Network
- ⚠️ nvidia-container-toolkit (Este no lo incluye 👍 )

## Descarga e instalación

Descarga e instalación de CUDA para Linux [NVIDIA CUDA Installation Guide for Linux](#)

Instalación de drivers mediante `ubuntu-drivers` [nvidia-drivers-installation](#)

## Ubuntu 24.04

```
sudo apt update && sudo apt install -y build-essential

wget https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2404/x86_64/cuda-ubuntu2404.pin
sudo mv cuda-ubuntu2404.pin /etc/apt/preferences.d/cuda-repository-pin-600
wget https://developer.download.nvidia.com/compute/cuda/12.8.0/local_installers/cuda-repo-ubuntu2404-12-8-local_12.8.0-570.86.10-1_amd64.deb
sudo dpkg -i cuda-repo-ubuntu2404-12-8-local_12.8.0-570.86.10-1_amd64.deb
sudo cp /var/cuda-repo-ubuntu2404-12-8-local/cuda-*-keyring.gpg /usr/share/keyrings/
```

## WSL 2

```
sudo apt update && sudo apt install -y build-essential

wget https://developer.download.nvidia.com/compute/cuda/repos/wsl-ubuntu/x86_64/cuda-wsl-ubuntu.pin
sudo mv cuda-wsl-ubuntu.pin /etc/apt/preferences.d/cuda-repository-pin-600
wget https://developer.download.nvidia.com/compute/cuda/12.8.0/local_installers/cuda-repo-wsl-ubuntu-12-8-local_12.8.0-1_amd64.deb
sudo dpkg -i cuda-repo-wsl-ubuntu-12-8-local_12.8.0-1_amd64.deb
sudo cp /var/cuda-repo-wsl-ubuntu-12-8-local/cuda-*-keyring.gpg /usr/share/keyrings/
```

Tras la actualización de paquetes podemos instalar, instalamos con `apt`, es posible que se requiera un reinicio.

```
sudo apt-get update
```

```
# Repositorio oficial de nvidia
```

```
sudo apt-get -y install cuda-toolkit-12-8
```

```
# export PATH="/usr/local/cuda-12.8/bin:$PATH"
```

```
# export LD_LIBRARY_PATH="/usr/local/cuda-12.8/lib64:$LD_LIBRARY_PATH"
```

```
# Repositorio oficial de ubuntu (es más sencillo instalar nvcc en WSL)
```

```
sudo apt-get -y install nvidia-cuda-toolkit
```

## Comprobar la instalación

```
nvcc --version
```

```
ls /usr/local/ | grep cuda
```

```
cd /etc/alternatives/cuda
```

```
realpath $(pwd) # Con realpath podéis resolver todos los enlaces simbólicos
```

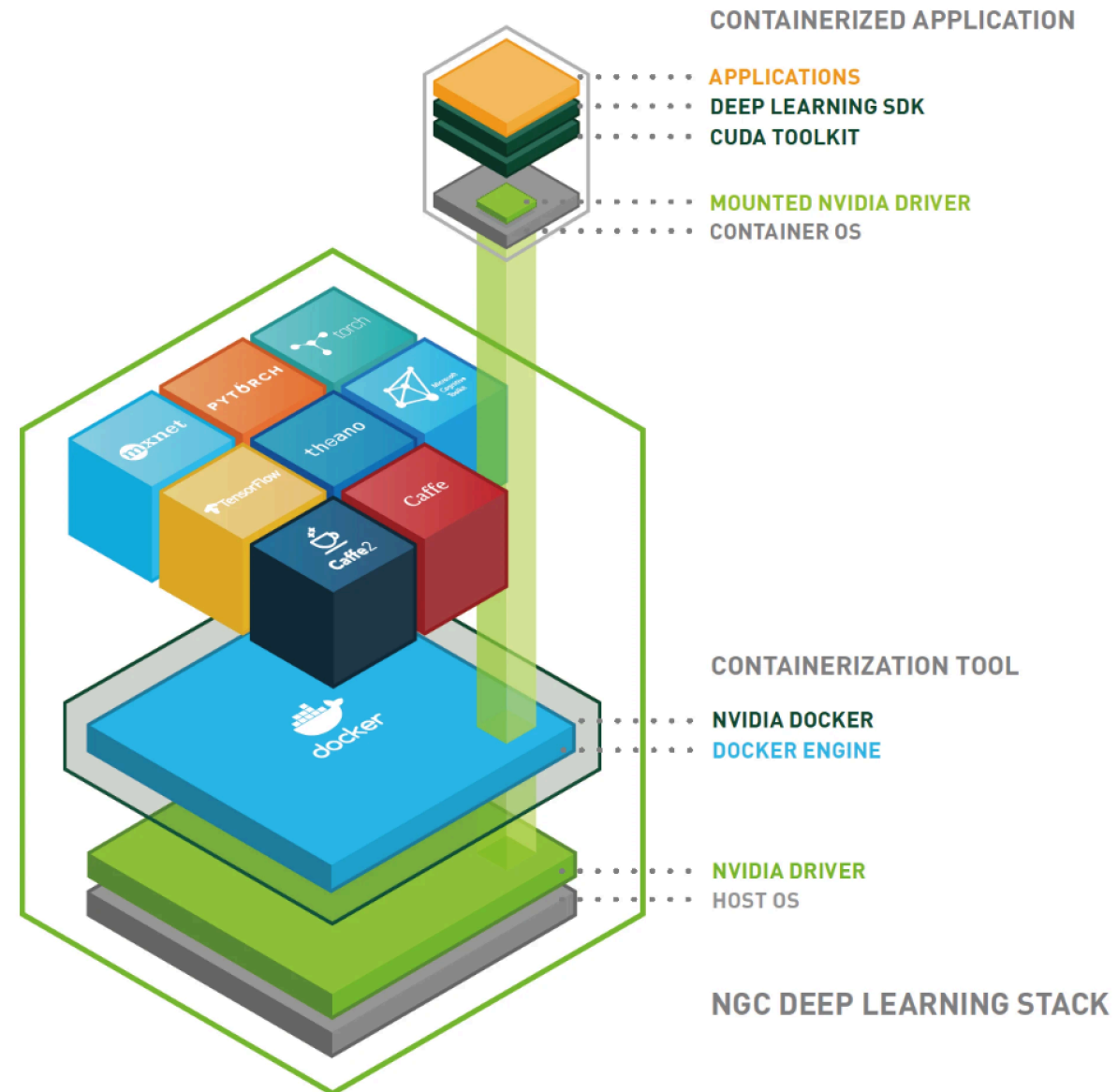


Descarga e instalación de CUDA para Windows [CUDA Installation Guide for Microsoft Windows](#)

Viene con la instalación del `container-toolkit` en el WSL predeterminado.

Instalación: Siguiente, siguiente, siguiente.

# Docker NVIDIA



# Instalación del runtime para GPUs de nvidia

Descargar repositorio de `nvidia-container-toolkit` [container-toolkit](#)

```
curl -fsSL https://nvidia.github.io/libnvidia-container/gpgkey | sudo gpg --dearmor -o /usr/share/keyrings/nvidia-container-toolkit-keyring.gpg \
&& curl -s -L https://nvidia.github.io/libnvidia-container/stable/deb/nvidia-container-toolkit.list | \
sed 's#deb https://#deb [signed-by=/usr/share/keyrings/nvidia-container-toolkit-keyring.gpg] https://#g' | \
sudo tee /etc/apt/sources.list.d/nvidia-container-toolkit.list
```

```
sudo sed -i -e '/experimental/ s/^#//g' /etc/apt/sources.list.d/nvidia-container-toolkit.list
sudo apt-get update
sudo apt-get install -y nvidia-container-toolkit
sudo nvidia-ctk runtime configure --runtime=docker
sudo systemctl restart docker
```

## Ejemplo básico

```
docker run --gpus all --rm -ti nvcr.io/nvidia/pytorch:25.01-py3
# Configuración de las Gráficas
docker run --gpus 2 ...
docker run --gpus "device=1,2" ...
docker run --gpus "device=UUID-ABCDEF,1" ...

docker run --gpus all --ipc=host --ulimit memlock=-1 --ulimit stack=67108864 --rm -ti nvcr.io/nvidia/pytorch:25.01-py3
```

## Errores comunes

```
docker run --gpus all -it --rm nvcr.io/nvidia/pytorch:25.01-py3  
> docker: Error response from daemon: could not select device driver "" with capabilities: [[gpu]].
```

Hay que revisar la instalación del `nvidia-container-toolkit`



```
1 #      Crear el contenedor
2 #      |      Todas las GPUs
3 #      |      |      Modo interactivo
4 #      |      |      |      Eliminar le contenedor despues de usarlo
5 #      |      |      |      |      permisos del usuario y grupo
6 #      |      |      |      |      |      Imagen a ejecutar
7 #      |      |      |      |      |
8 #      v      v      v      v      v
9 docker run --gpus all -ti --rm -u $(id -u):$(id -g) nvcr.io/nvidia/pytorch:24.04-py3
```

Puede aparecer esta advertencia "groups: cannot find name for group ID 1000 I have no name!", puedes ignorarla

# Docker compose

Despliegue de soluciones con docker compose (orquestración).

```
# Obtener más información  
docker compose [orden] --help
```


```
# Construir los contenedores  
docker compose build [servicio]
```

```
# Parar los servicios y eliminar los contenedores  
docker compose down [servicio]
```

```
# Levantar el servicio  
docker compose up [servicio] -d # `-d` para segundo plano
```

# NVIDIA CATALOG

## Catálogo de contenedores de NVIDIA

 NGC Catalog

Welcome Guest

Explore Catalog

Collections

Containers

Helm Charts

Models

Resources

The NGC catalog hosts containers for AI/ML, metaverse, and HPC applications and are performance-optimized, tested, and ready to deploy on GPU-powered on-prem, cloud, and edge systems.

Search containers...

Most Popular

Use Case

☐ Recommendation

☐ Drug Discovery

☐ Facial Landmark Estimation

☐ Natural Language Processing

6

2

1

1

NVIDIA Platform

☐ Metropolis

☐ Triton Inference Server

☐ Clara AGX

☐ Merlin

☒ TensorFlow

☐ DeepStream

☐ Isaac

☐ Metropolis Microservices

☐ Morpheus

☒ RAPIDS

☐ NeMo

☐ Maxine

☐ Modulus

☐ Mosaic

11

11

9

8

8

7

7

6

6

6

5

4

4

4

NVIDIA NIM

☐ NVIDIA NIM

2

NVIDIA Enterprise Platforms

☐ NVIDIA AI Enterprise Supported

☐ NVIDIA AI Enterprise Essentials

16

11


Displaying 28 containers


PyTorch

RAPIDS

TensorFlow

Clear filters


 PyTorch


Accelerated with  


PyTorch

PyTorch is a GPU accelerated tensor computational framework. Functionality can be extended with common Python...

NVIDIA AI Enterprise Supported


 TensorFlow


Accelerated with  


TensorFlow

TensorFlow is an open source platform for machine learning. It provides comprehensive tools and libraries in a...


NVIDIA AI Enterprise Supported

 RAPIDS

Accelerated with  



RAPIDS Notebooks


The RAPIDS suite of software libraries gives you the freedom to execute end-to-end data science and analytics...

 NVIDIA AI Workbench


PyTorch for AI Workbench

PyTorch - AI Workbench Default Container (Beta)

 PyTorch

Accelerated with  


RAPIDS

Accelerated with  


32



# Descargar imágenes

```
docker pull nvcr.io/nvidia/pytorch:25.01-py3      # 6 minutos aproximadamente
docker pull nvcr.io/nvidia/tensorflow:25.01-tf2-py3
docker pull nvcr.io/nvidia/rapidsai/notebooks:24.12-cuda12.5-py3.12

docker pull ollama/ollama
docker pull ghcr.io/open-webui/open-webui:main
```

```
# https://hub.docker.com/r/nvidia/cuda
docker pull nvidia/cuda:11.8.0-base-ubuntu22.04
```

# NVIDIA-Pytorch

```
services:
  runner_pytorch:
    container_name: ${PROJECT_NAME}_runner_pytorch
    image: nvcr.io/nvidia/pytorch:25.01-py3
    # build:
    #   context: ./Apps/runner-pytorch
    #   dockerfile: Dockerfile
    command: bash
    stdin_open: true
    tty: true
    ipc: host # Para compartir memoria
    environment:
      - NVIDIA_VISIBLE_DEVICES=all
      - NVIDIA_DRIVER_CAPABILITIES=all
    volumes:
      - ./Apps/runner-pytorch/workspace_pytorch:/workspace_pytorch
    runtime: nvidia
    deploy:
      resources:
        reservations:
          devices:
            - driver: nvidia
              count: all
              capabilities:
                - gpu
```

```

1 2025-02-12 21:26:09
2 2025-02-12 21:26:09 =====
3 2025-02-12 21:26:09 == PyTorch ==
4 2025-02-12 21:26:09 =====
5 2025-02-12 21:26:09
6 2025-02-12 21:26:09 NVIDIA Release 25.01 (build 134983853)
7 2025-02-12 21:26:09 PyTorch Version 2.6.0a0+ecf3bae
8 2025-02-12 21:26:09 Container image Copyright (c) 2025, NVIDIA CORPORATION & AFFILIATES. All rights reserved.
9 2025-02-12 21:26:09 Copyright (c) 2014-2024 Facebook Inc.
10 2025-02-12 21:26:09 Copyright (c) 2011-2014 Idiap Research Institute (Ronan Collobert)
11 2025-02-12 21:26:09 Copyright (c) 2012-2014 Deepmind Technologies (Koray Kavukcuoglu)
12 2025-02-12 21:26:09 Copyright (c) 2011-2012 NEC Laboratories America (Koray Kavukcuoglu)
13 2025-02-12 21:26:09 Copyright (c) 2011-2013 NYU (Clement Farabet)
14 2025-02-12 21:26:09 Copyright (c) 2006-2010 NEC Laboratories America (Ronan Collobert, Leon Bottou, Iain Melvin, Jason Weston)
15 2025-02-12 21:26:09 Copyright (c) 2006 Idiap Research Institute (Samy Bengio)
16 2025-02-12 21:26:09 Copyright (c) 2001-2004 Idiap Research Institute (Ronan Collobert, Samy Bengio, Johnny Mariethoz)
17 2025-02-12 21:26:09 Copyright (c) 2015 Google Inc.
18 2025-02-12 21:26:09 Copyright (c) 2015 Yangqing Jia
19 2025-02-12 21:26:09 Copyright (c) 2013-2016 The Caffe contributors
20 2025-02-12 21:26:09 All rights reserved.
21 2025-02-12 21:26:09
22 2025-02-12 21:26:09 Various files include modifications (c) NVIDIA CORPORATION & AFFILIATES. All rights reserved.
23 2025-02-12 21:26:09
24 2025-02-12 21:26:09 This container image and its contents are governed by the NVIDIA Deep Learning Container License.
25 2025-02-12 21:26:09 By pulling and using the container, you accept the terms and conditions of this license:
26 2025-02-12 21:26:09 https://developer.nvidia.com/ngc/nvidia-deep-learning-container-license
27 2025-02-12 21:26:10

```

```

1 root@bd282a32daa1:/workspace_pytorch# python test.py
2 Variable                                     Value
3 -----
4 torch.__version__                           2.6.0a0+ecf3bae40a.nv25.01
5 torch.cuda.is_available                     True
6 torch.cuda.current_device                   0
7 torch.cuda.device_count                     1
8 torch.cuda.get_device_name                  NVIDIA GeForce RTX 4060
9 torch.cuda.is_initialized                   True
10 torch.cuda.memory_allocated                 0
11 torch.cuda.memory_reserved                 0
12 torch.cuda.max_memory_allocated             0
13 torch.cuda.max_memory_reserved              0
14 torch.backends.cpu.get_cpu_capability       AVX512
15 torch.backends.cudnn.is_available           True
16 torch.backends.mkl.is_available             True
17 torch.backends.mkldnn.is_available          True
18 torch.backends.mps.is_available             False
19 torch.backends.openmp.is_available          True

```

# NVIDIA-TensorFlow

```
services:
  runner_tensorflow:
    container_name: ${PROJECT_NAME}_runner_tensorflow
    # image: nvcr.io/nvidia/tensorflow:25.01-tf2-py3
    build:
      context: ./Apps/runner-tensorflow
      dockerfile: Dockerfile
    command: bash
    stdin_open: true
    tty: true
    ipc: host
    environment:
      - NVIDIA_VISIBLE_DEVICES=all
      - NVIDIA_DRIVER_CAPABILITIES=all
    volumes:
      - ./Apps/runner-tensorflow/workspace_tensorflow:/workspace_tensorflow
    runtime: nvidia
    deploy:
      resources:
        reservations:
          devices:
            - driver: nvidia
              count: all
              capabilities:
                - gpu
```

```

1 2025-02-12 21:37:24
2 2025-02-12 21:37:24 =====
3 2025-02-12 21:37:24 == TensorFlow ==
4 2025-02-12 21:37:24 =====
5 2025-02-12 21:37:24
6 2025-02-12 21:37:24 NVIDIA Release 25.01-tf2 (build 134984172)
7 2025-02-12 21:37:24 TensorFlow Version 2.17.0
8 2025-02-12 21:37:24 Container Image Copyright (c) 2025, NVIDIA CORPORATION & AFFILIATES. All rights reserved.
9 2025-02-12 21:37:24 Copyright 2017-2024 The TensorFlow Authors. All rights reserved.
10 2025-02-12 21:37:24
11 2025-02-12 21:37:24 Various files include modifications (c) NVIDIA CORPORATION & AFFILIATES. All rights reserved.
12 2025-02-12 21:37:24
13 2025-02-12 21:37:24 This container image and its contents are governed by the NVIDIA Deep Learning Container License.
14 2025-02-12 21:37:24 By pulling and using the container, you accept the terms and conditions of this license:
15 2025-02-12 21:37:24 https://developer.nvidia.com/ngc/nvidia-deep-learning-container-license
16 2025-02-12 21:37:24

```

```

1 Variable Value
2 -----
3 tf.__version__ 2.17.0
4 tf test is_built_with_cuda True
5 tf test is_gpu_available True
6 tf test gpu_device_name /device:GPU:0
7 tf config list_physical_devices (GPU) [PhysicalDevice(name='/physical_device:GPU:0', device_type='GPU')]
8 tf config experimental get_memory_info(0) {'current': 0, 'peak': 0}
9 tf config experimental get_memory_usage(0) 0
10 tf config experimental get_device_details(0) {'compute_capability': (8, 9), 'device_name': 'NVIDIA GeForce RTX 4060'}
11 ----
12 tf device_lib [name: "/device:CPU:0"
13 device_type: "CPU"
14 memory_limit: 268435456
15 locality {
16 }
17 incarnation: 16310524361366595393
18 xla_global_id: -1
19 , name: "/device:GPU:0"
20 device_type: "GPU"
21 memory_limit: 5556404224
22 locality {
23 bus_id: 1
24 links {
25 }
26 }
27 incarnation: 968707182923022416
28 physical_device_desc: "device: 0, name: NVIDIA GeForce RTX 4060, pci bus id: 0000:01:00.0, compute capability: 8.9"
29 xla_global_id: 416903419
30 ]

```

# NVIDIA-RAPIDSAI

La imagen `rapidsai/base` contiene una shell de `ipython` de manera predeterminada.

La imagen `rapidsai/notebooks` contiene el servidor de `JupyterLab` de manera predeterminada.

```
RAPIDS version
|      CUDA version
|      |      Python version
v      v      v
24.12-cuda12.5-py3.12
```



Name Modified

cudf	last mo.
cugraph	last mo.
cuml	last mo.
cuspatial	last mo.



Launcher



Notebook



Python 3  
(ipykernel)



Console



Python 3  
(ipykernel)



Other



Terminal



Text File



Markdown File



Python File



Show Contextual  
Help

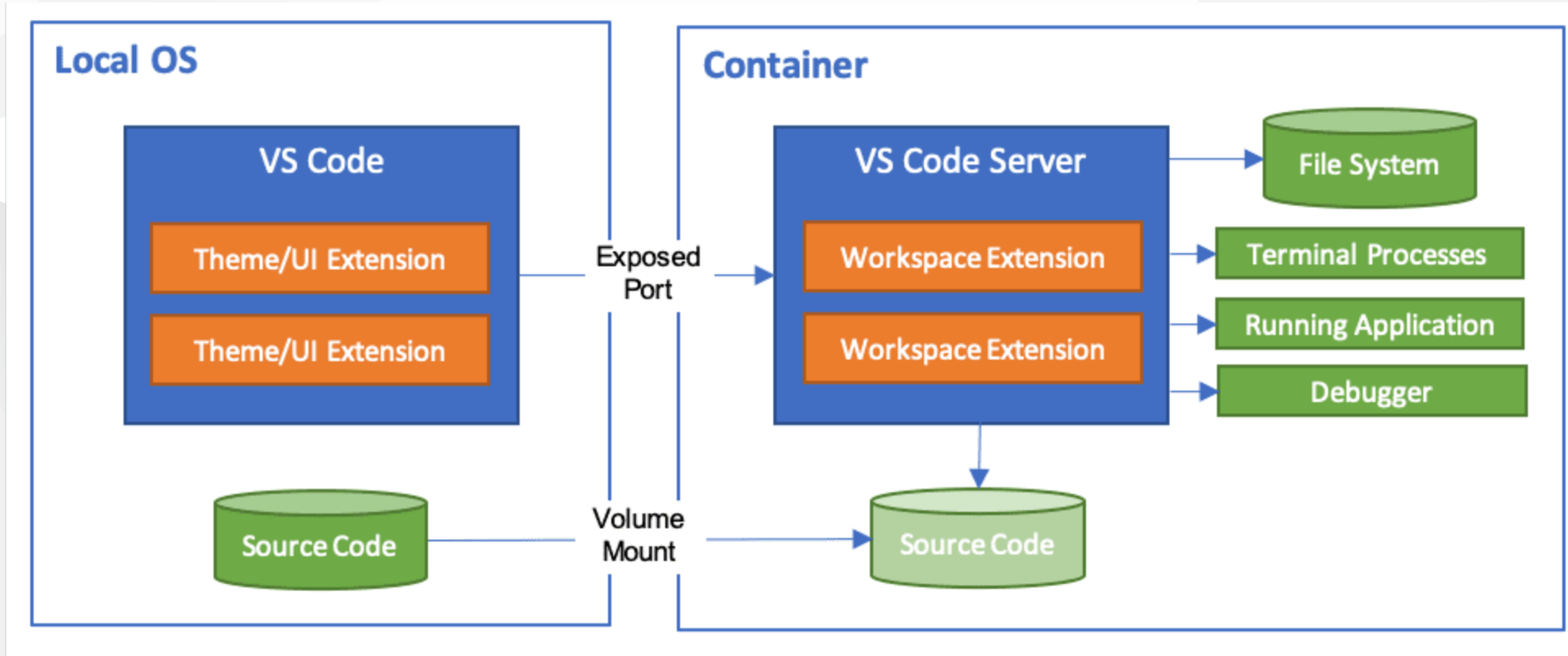
```

services:
  runner_rapidsai:
    container_name: ${PROJECT_NAME}_runner_rapidsai-notebook
    image: nvcr.io/nvidia/rapidsai/notebooks:24.12-cuda12.5-py3.12
    #build:
    # context: ./Apps/runner-3-rapidsai
    # dockerfile: Dockerfile
    ports:
      - 8888:8888
    stdin_open: true
    tty: true
    ipc: host # Memoria compartida
    environment:
      - NVIDIA_VISIBLE_DEVICES=all
      - NVIDIA_DRIVER_CAPABILITIES=all
      # - EXTRA_CONDA_PACKAGES=jq      # Paquetes extra de conda a instalar
      # - CONDA_TIMEOUT=5                # Espera de 5 segundos tras instalar paquetes de conda.
      # - EXTRA_PIP_PACKAGES=tabulate  # Paquetes extra de pip a instalar
      # - PIP_TIMEOUT=5                  # Espera de 5 segundos tras instalar paquetes de pip.
    shm_size: "1gb"
    ulimits:
      memlock: -1      # Permitir el bloqueo ilimitado de la memoria
      stack: 67108864  # Tamaño de la pila en bytes
    volumes:
      - ./Apps/runner-3-rapidsai/workspace_rapidsai:/workspace_rapidsai
    runtime: nvidia
    deploy:
      resources:
        reservations:
          devices:
            - driver: nvidia
              count: 1
              capabilities: [gpu]

```



# DEV Container



# OLLAMA

Ollama es un gestor de modelos LLM que permite descargar, ejecutar y desplegar modelo LLM fácilmente mediante un servidor que distribuye una [API](#).

Esta tecnología nos permite simplificar la distribución de modelos para distintos usos.

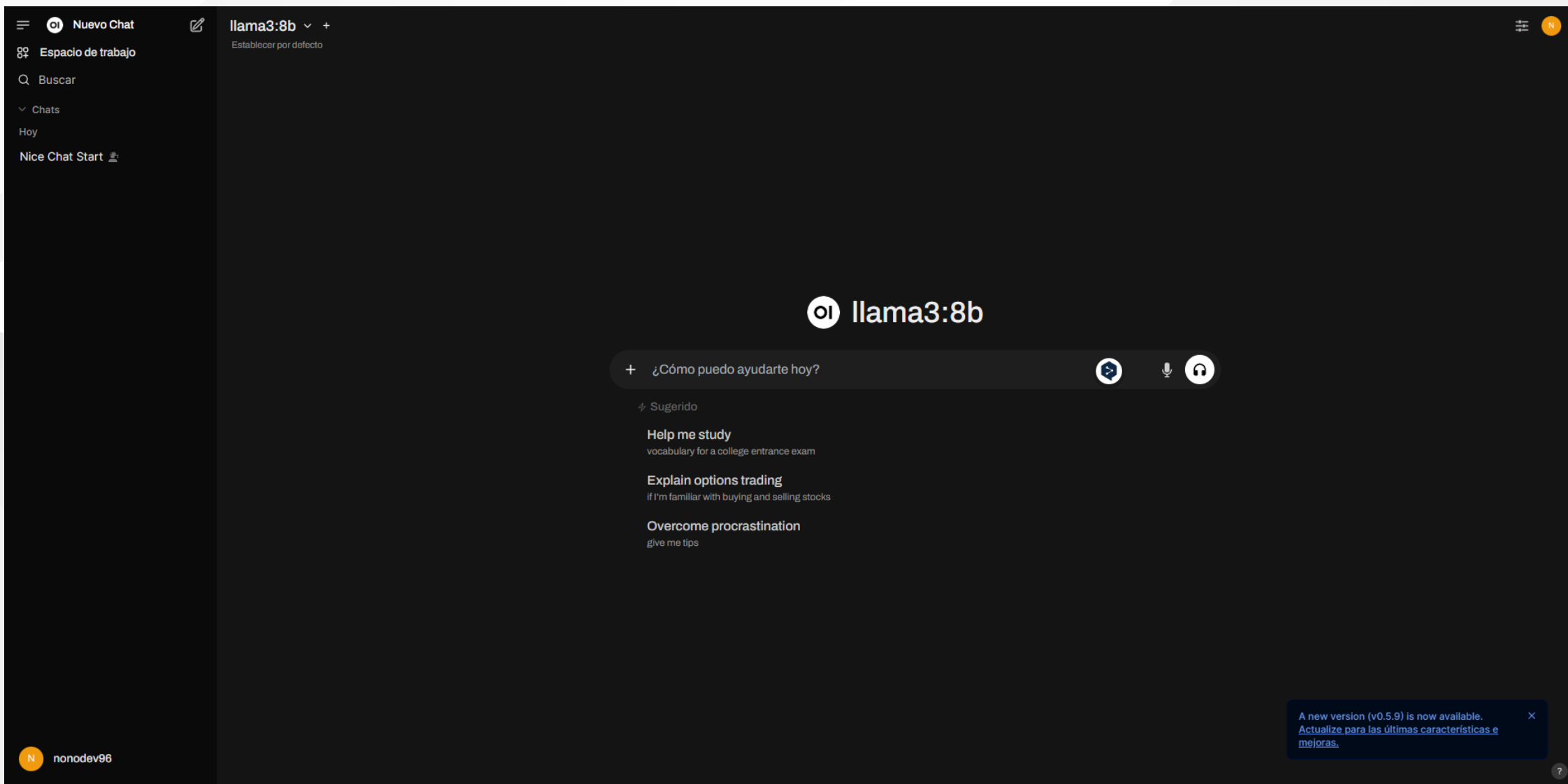


# Open WebUI

Open-WebUI es una interfaz web de código abierto para interactuar con modelos de inteligencia artificial locales y en la nube. Suelen usarse con modelos LLM como Llama, Mistral o GPT a través de servidores como Ollama o LM Studio.

[@open-webui/open-webui](#)

[openwebui.com](https://openwebui.com)



## Ejemplo de docker compose de un chat-gpt propio

```
services:
  open-webui:
    image: ghcr.io/open-webui/open-webui:main
    container_name: ${PROJECT_NAME}_open-webui
    restart: unless-stopped
    volumes:
      - local-open-webui:/app/backend/data
    depends_on:
      - ollama
    ports:
      - ${OPEN_WEBUI_PORT-3000}:8080
    environment:
      - "OLLAMA_BASE_URL=http://ollama:11434"
    extra_hosts:
      - host.docker.internal:host-gateway
```

```
ollama:
  image: ollama/ollama:latest
  container_name: ${PROJECT_NAME}_ollama
  restart: unless-stopped
  tty: true
  volumes:
    - local-ollama:/root/.ollama
  pull_policy: always
  # GPU support
  runtime: nvidia
  deploy:
    resources:
      reservations:
        devices:
          - driver: nvidia
            count: 1
            capabilities:
              - gpu
volumes:
  local-ollama:
    external: false
  local-open-webui:
    external: false
```

# Traefik

Para traefik debemos añadir la redirección al servicio, con ubuntu/debian `sudo nano /etc/hosts` y para Windows abrir el fichero `C:\Windows\System32\drivers\etc\hosts` con el editor de texto dando permisos de administrador y añadir los siguientes DNS.

```
# Añadimos el host  
127.0.0.1 chat.nonodev96.dev
```

# Referencias

- [CUDA Toolkit](#)
  - [CUDA GUIDE Windows](#)
  - [CUDA GUIDE Linux](#)
- [NVIDIA Container Toolkit](#)
- [Catálogo de contenedores de NVIDIA](#)
- [VSCODE Extensión Docker](#)
- [cuDNN](#)
- [cuBLAS](#)
- [cuSPARSE](#)
- [OPEN WEB UI](#)
- [ollama](#)



## Consideraciones en Windows

Puedes tener distintos docker en windows, asegúrate de estar usando el adecuado, pues la imágenes ocuparan mucha memoria

```
C:\Users\nono_>wsl.exe --list
Distribuciones de subsistema de Windows para Linux:
Ubuntu (predeterminado)
Ubuntu-24.04
docker-desktop # <-- Puedes desactivarlo, pero pierdes el puente del runtime
```

```
C:\Users\nono_>wsl.exe --shutdown
```