# MLOps, Servicios Cloud y Docker

## Arquitecturas e Infraestructura para la Inteligencia Artificial

Manuel Benavent-Lledó <mbenavent@dtic.ua.es>
David Mulero-Pérez <dmulero@dtic.ua.es>
José García-Rodríguez <jgarcia@dtic.ua.es>

# CONTENIDO

- Ciclo MLOps
- Servicios Cloud
- Introducción a Docker
- Uso de Docker

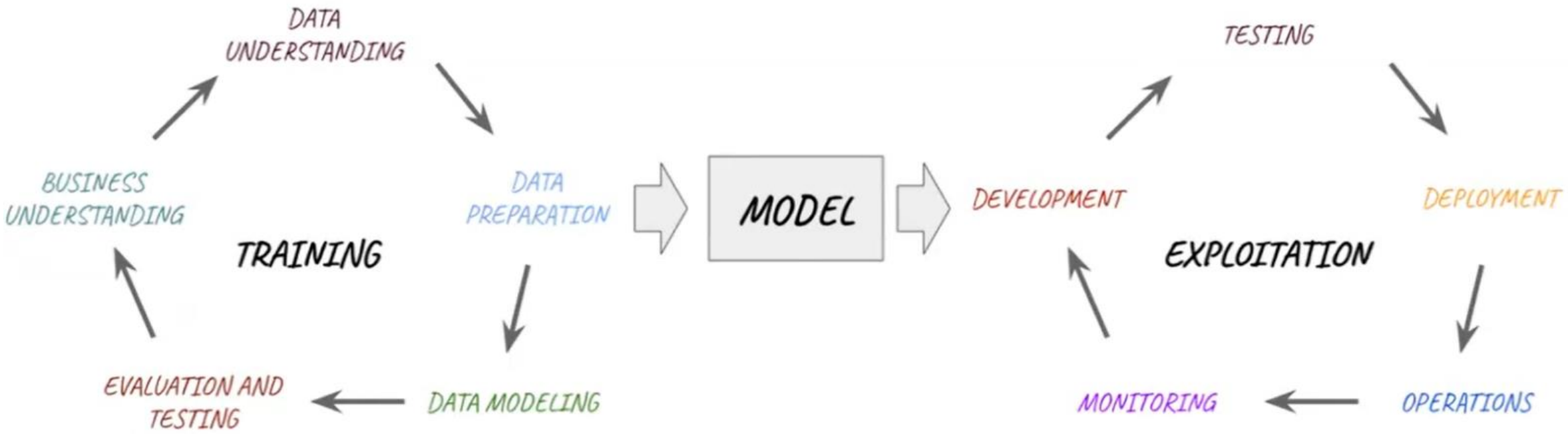
# Ciclo MLOps

## ¿Por qué?

- Al igual que en el desarrollo software existe DevOps, es necesario un sistema para automatizar los procesos
- Al desplegar un modelo no acaba el trabajo
  - Hay que re-entrenar si baja la precisión
  - Hay nuevos datos
  - Nuevas clases
- Es un proceso iterativo: utilizando la metodología *Agile* podemos hacer ciclos cortos e ir entregando valor al cliente

# Ciclo MLOps

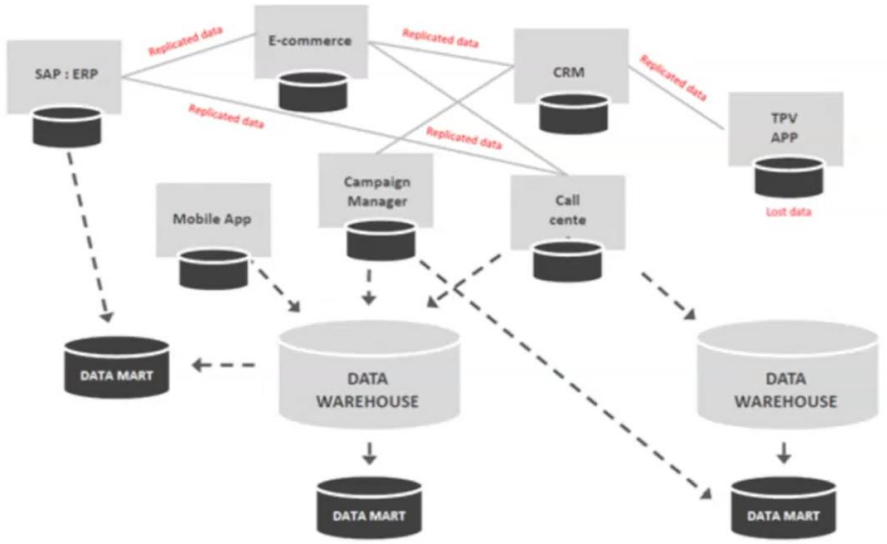
## Representación en 2 ciclos



# Ciclo MLOps

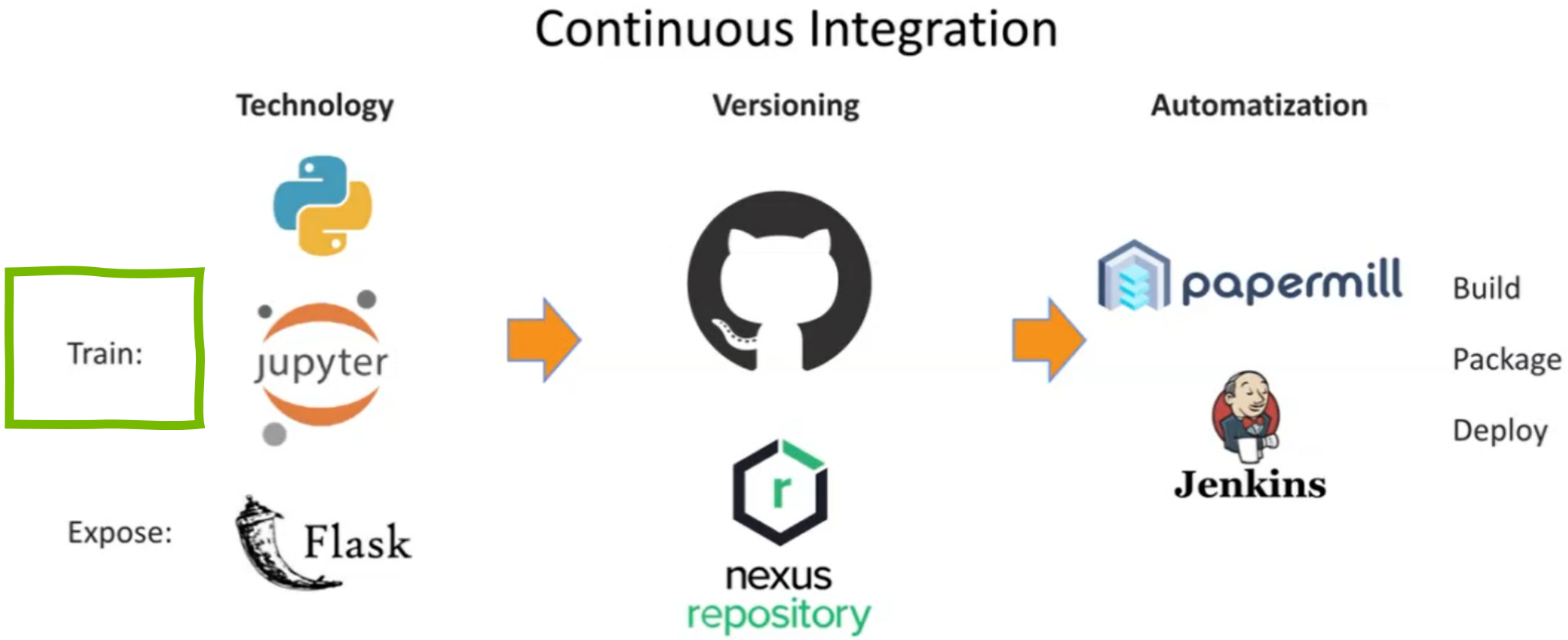
## Objetivos

- Integración continua y automatización: eliminar tareas manuales
  - Ej: AutoML nos permite automatizar la limpieza de datos, selección de modelos, etc.
- Organización en equipos multidiscliplinares para evitar retrasos y problemas de comunicación
- Agrupación de los datos para evitar duplicidad
- Entrenamiento de modelos:
  - Control de vresiones
  - Automatizar commits, validaciones y testing de las diferentes partes



# Ciclo MLOps

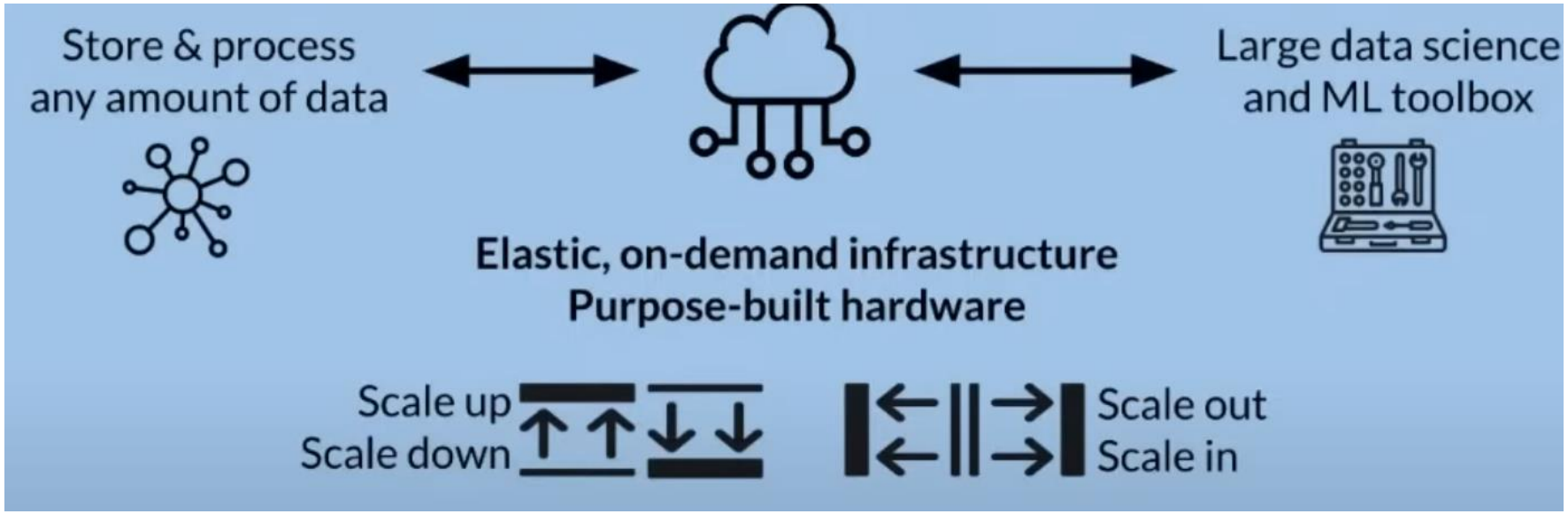
## Integración continua



# Servicios Cloud

## Gestión de modelos en la nube

- Los recursos locales son limitados, las arquitecturas en la nube nos permiten escalar
- Acceso flexible con simplemente conexión a internet
- Mejora en la colaboración trabajando en equipo
- Pago por uso: solo pagamos cuando entrenamos (aunque los recursos pueden llegar a ser muy caros)



# Servicios Cloud

## Escalabilidad

- Escalado vertical (up/down)
  - Aumentar la potencia de un recurso individualmente
  - Ej: CPU/GPU más potente, más RAM, más almacenamiento, etc
  - Limitaciones en la potencia de los servidores

- Escalado horizontal (in/out)
  - Crear más instancias de un mismo recurso
  - Ej: Más servidores para distribuir el entrenamiento
  - Requiere preparar los modelos para soportar un entrenamiento de este tipo
  - Tensorflow y Pytorch soportan entrenamiento distribuido

# Servicios Cloud

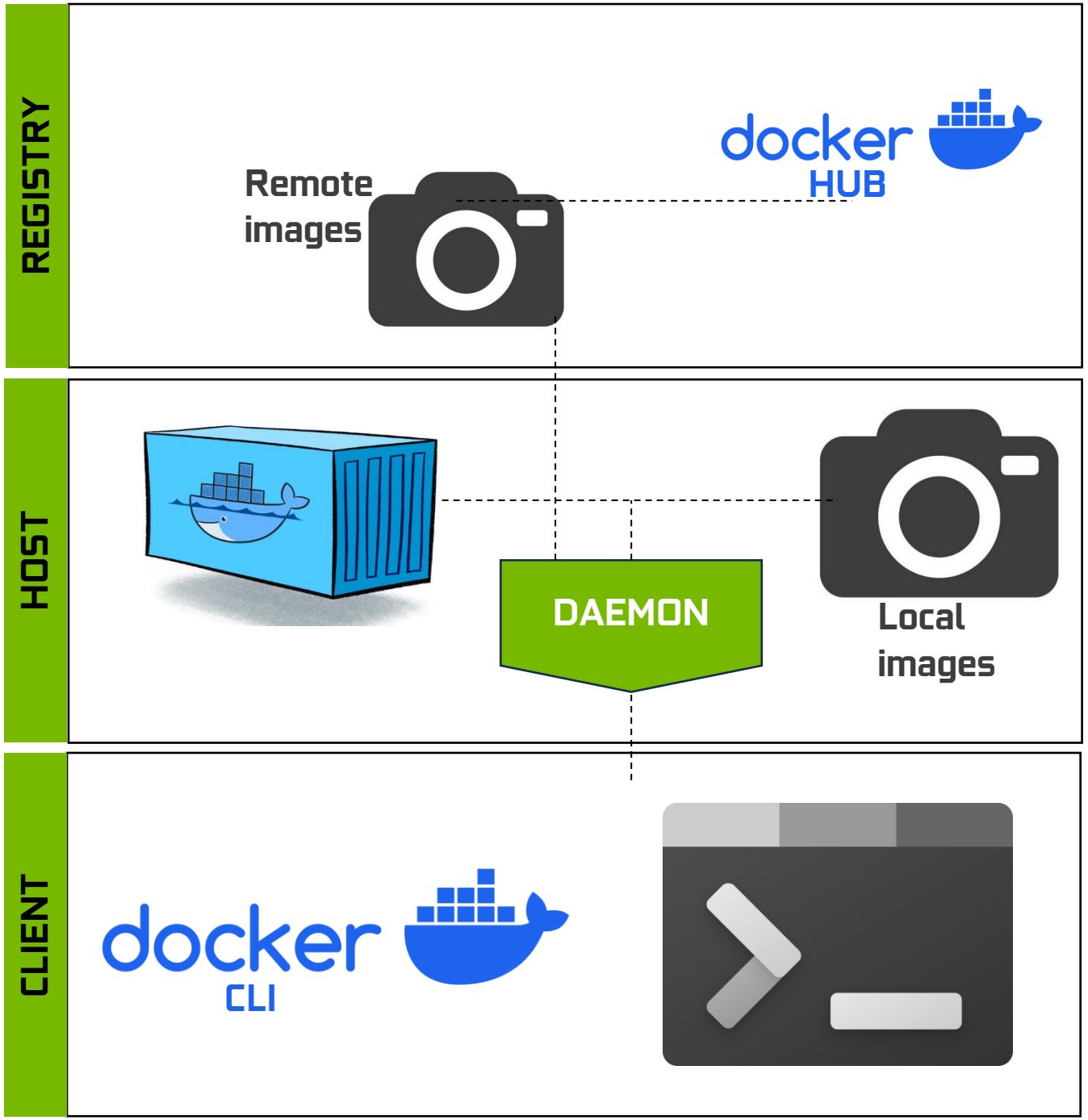
## Proveedores principales



# Introducción a Docker

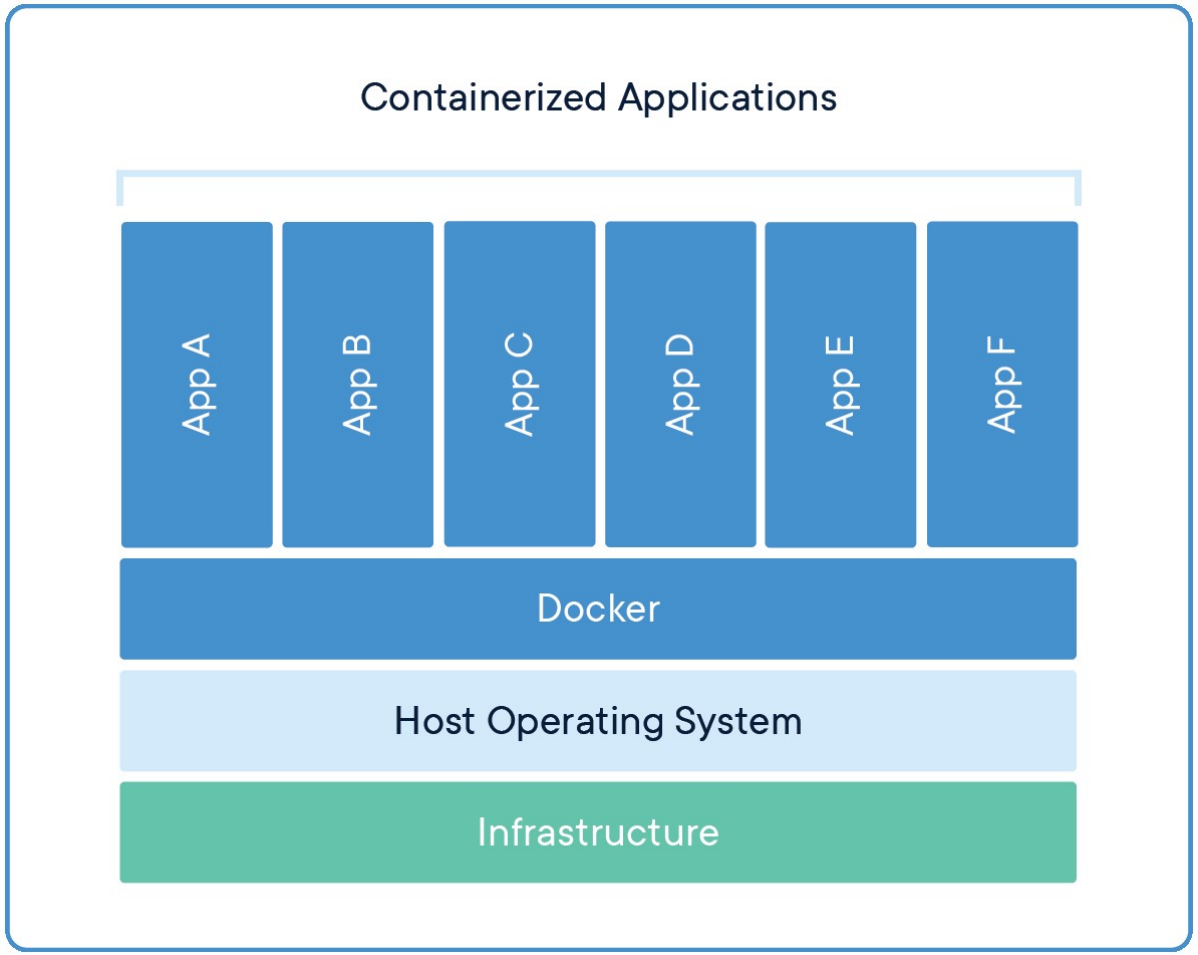
## ¿En qué consiste?

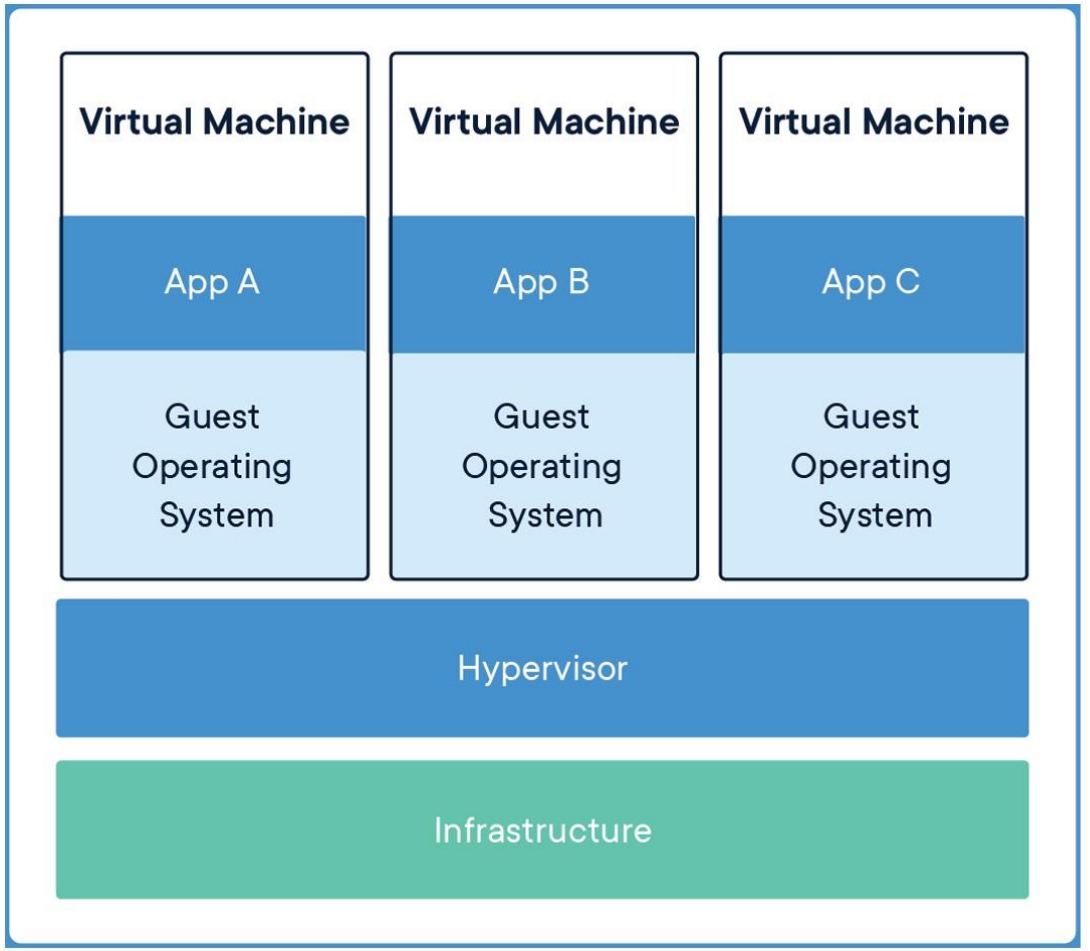
- Sistema Operativo para contenedores
- Contenedor
  - Empaqueta software de forma ligera
  - Se crea a partir de una imagen
    - Descargada de internet (docker hub)
    - A partir de un Dockerfile



# Introducción a Docker

## Ventajas sobre las VMs




**Cuidado con Windows!!**

# Introducción a Docker

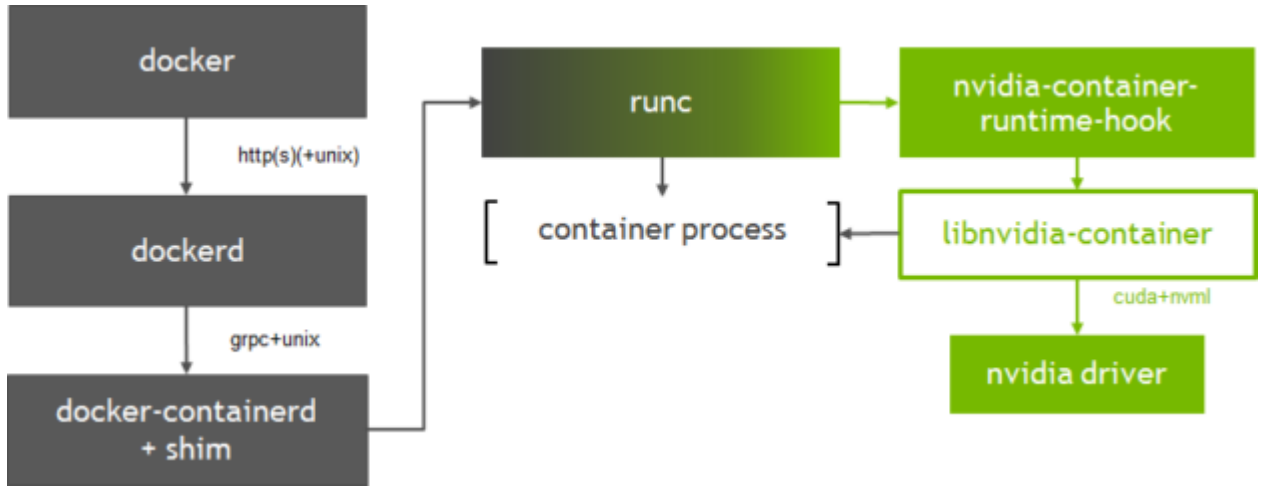
## Uso en ML/DL

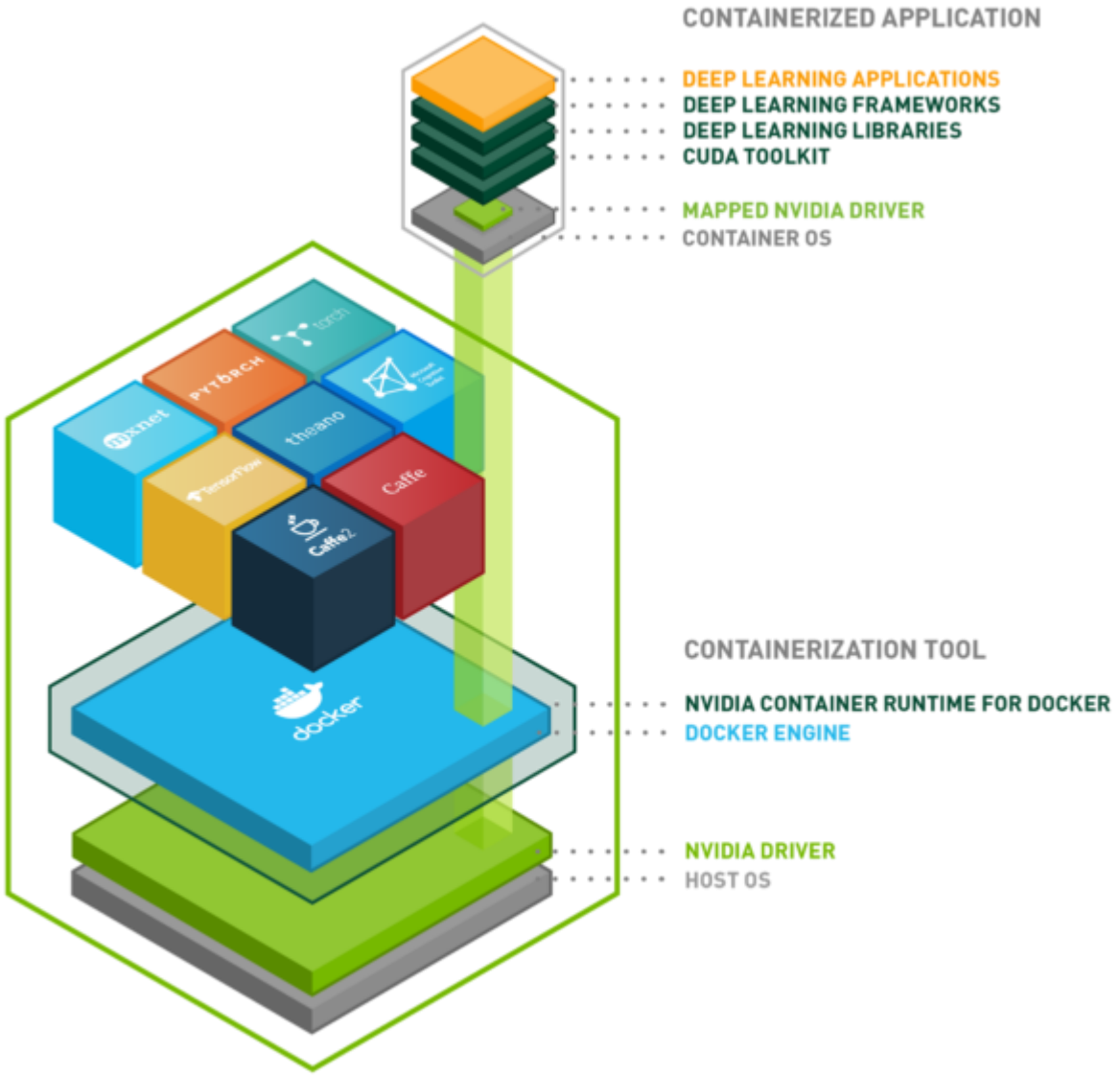
- Entornos aislados y reproducibles
- Elimina conflictos de dependencias
- Portabilidad
- Gestión de Recursos controlada
- Fácil distribución
- Rápido despliegue y escalabilidad
- Manejo de versiones y rollbacks
- Automatización con Docker Compose

# Introducción a Docker

## NVIDIA Container Toolkit

- Uso de Docker con GPUs NVIDIA
- Solo Linux

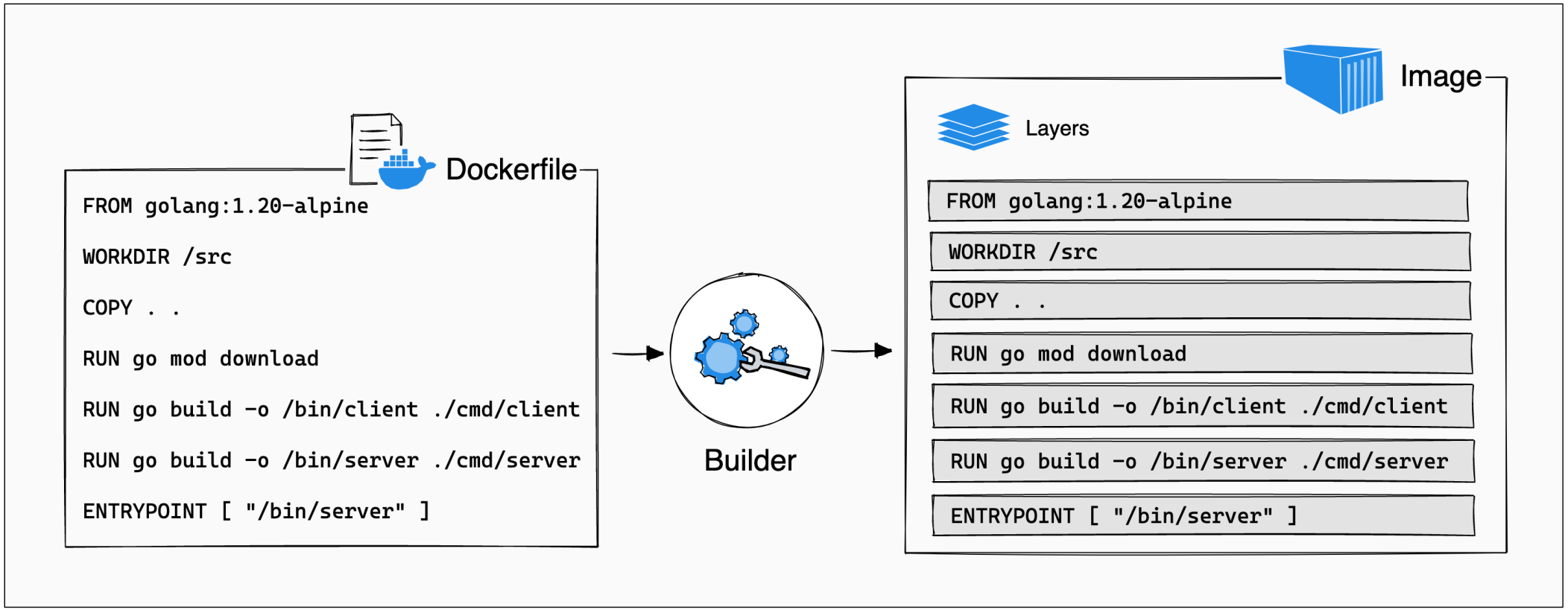


# Uso de Docker

## Dockerfile

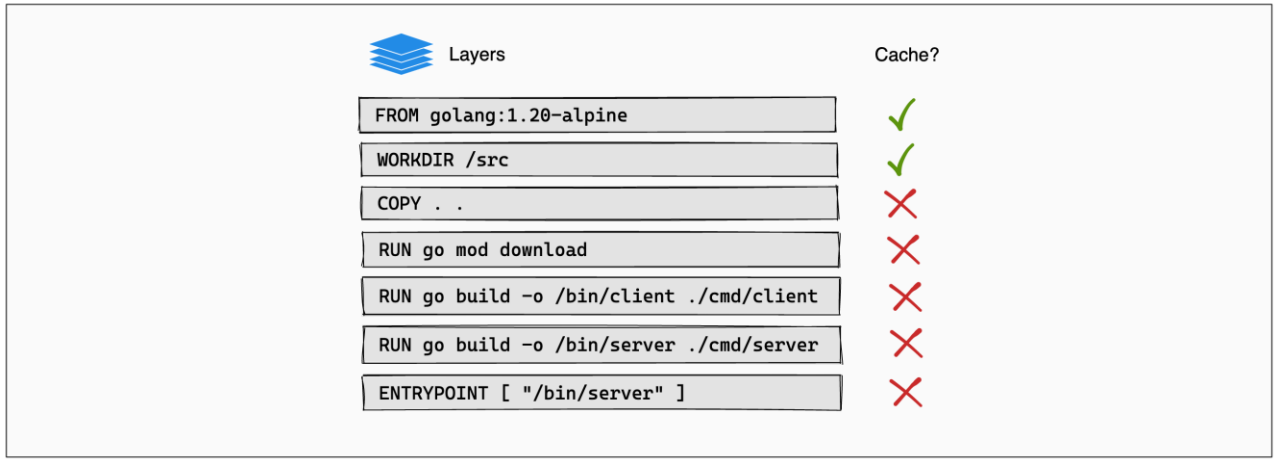
- Documento con instrucciones  para construir una imagen nueva

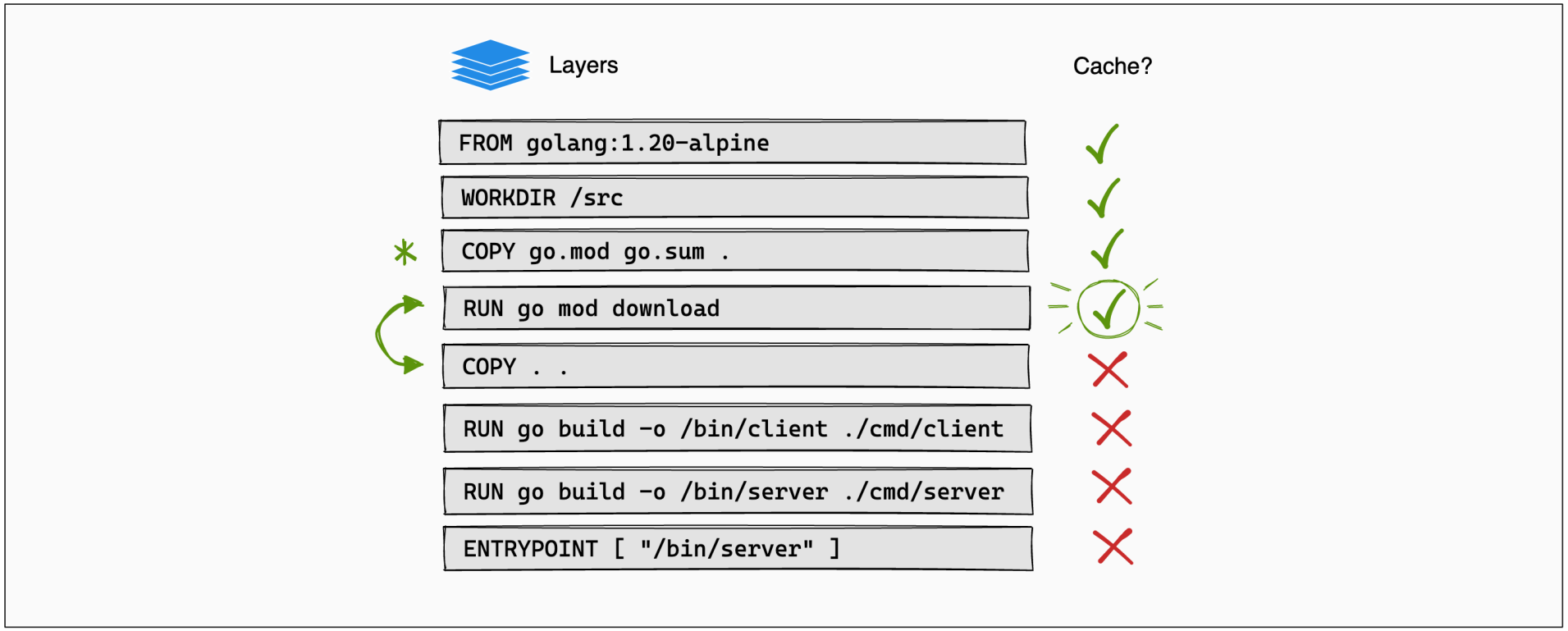


# Uso de Docker

## Dockerfile

El orden de las capas es relevante!!



# Uso de Docker

## Ejecución de contenedores

```
#!/bin/bash

export containerName=nombreProyecto_$USER

docker run
  -d # detatch mode
  --gpus '"device=0“’ # GPU device: 0, 1, all
  --rm # remove container once it has stopped
  -it # -i: interactive | -t: pseudo-TTY (-it: make it look like a regular terminal)
  --volume="/path_host:/path_container:rw" # mount a volume with read-write permission
  --volume="/path_host:/path_container:ro" # mount a volume with read-only permission
  --workdir="/workspace" # set workspace directory
  --shm-size=16g # increase shared memory size (default 64MB)
  --memory=16g # limit RAM memory
  --name $containerName # set container name
  repo/image:version bash
```

# Uso de Docker

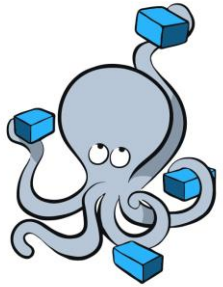
## Entorno gráfico y red

```
#!/bin/bash

export containerName=nombreProyecto_$USER
sleep 3 && \
        xhost +local:`docker inspect --format='{{ .Config.Hostname
}}' $containerName` >/dev/null 2>&1 &
docker run […]
        --volume=$HOME/.Xauthority:/root/.Xauthority:ro \
        --net=host \
        --env="DISPLAY" \
        --name $containerName \
        repo/image:version bash
```

# Uso de Docker

## Docker Compose



docker Compose

- Multi-contenedor
- Orquestación simplificada
  - Asegura dependencias correctas
- Archivos YAML de configuración
- Facilidad de despliegue

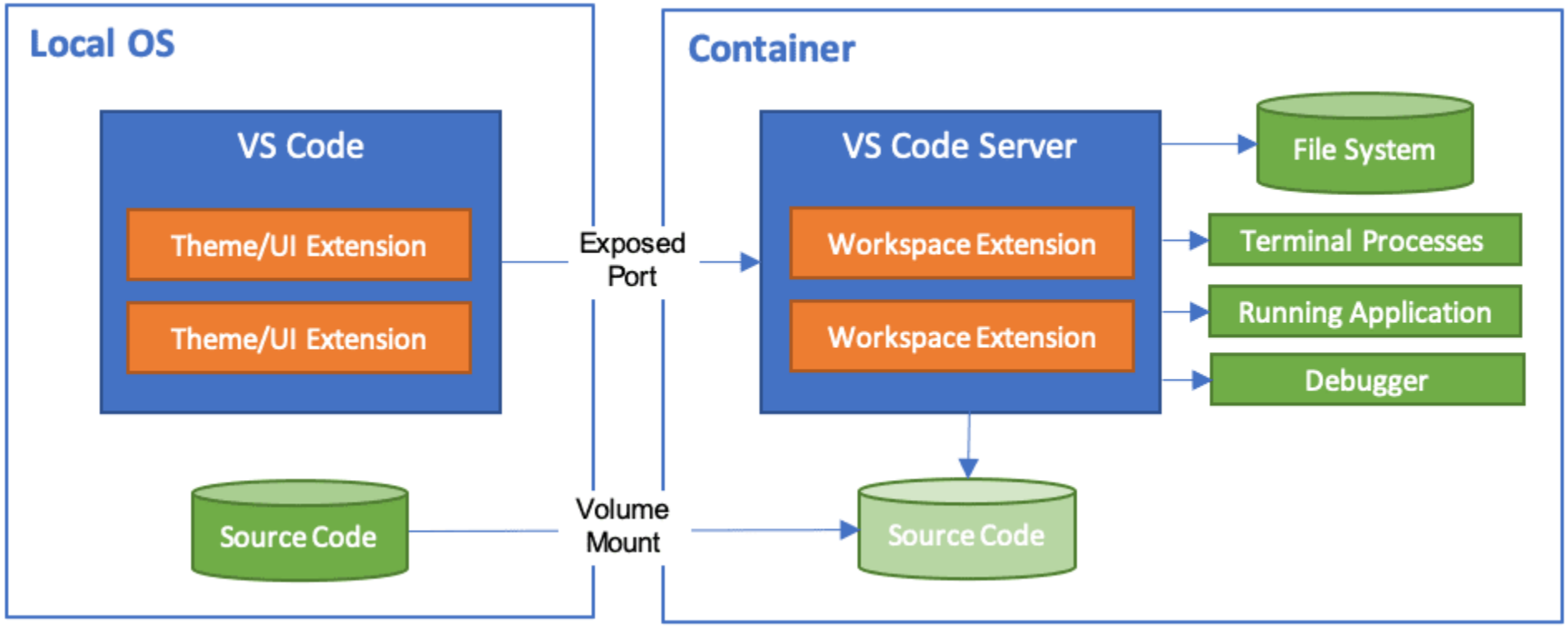
```yaml
services:
  web:
    build: .
    ports:
      - "5000:5000"
    volumes:
      - .:/code
  redis:
    image: redis
```

```
docker compose up [-d]
docker compose down
```

# Uso de Docker

## VSCode – DevContainers

- Conexión con VSCode : debug, jupyter notebooks, ...



# Uso de Docker

## Hugging Face Spaces

- Despliegue de aplicaciones en repositorios de Hugging Face
- FastAPI para publicar servicios RESTful
- Conectado a herramientas de automatización