En mi máquina funciona, pero ¿y en la tuya?

Introducción

Requisitos

Instalación

- Docker Engine (WSL 2 / Linux)
- Docker Compose

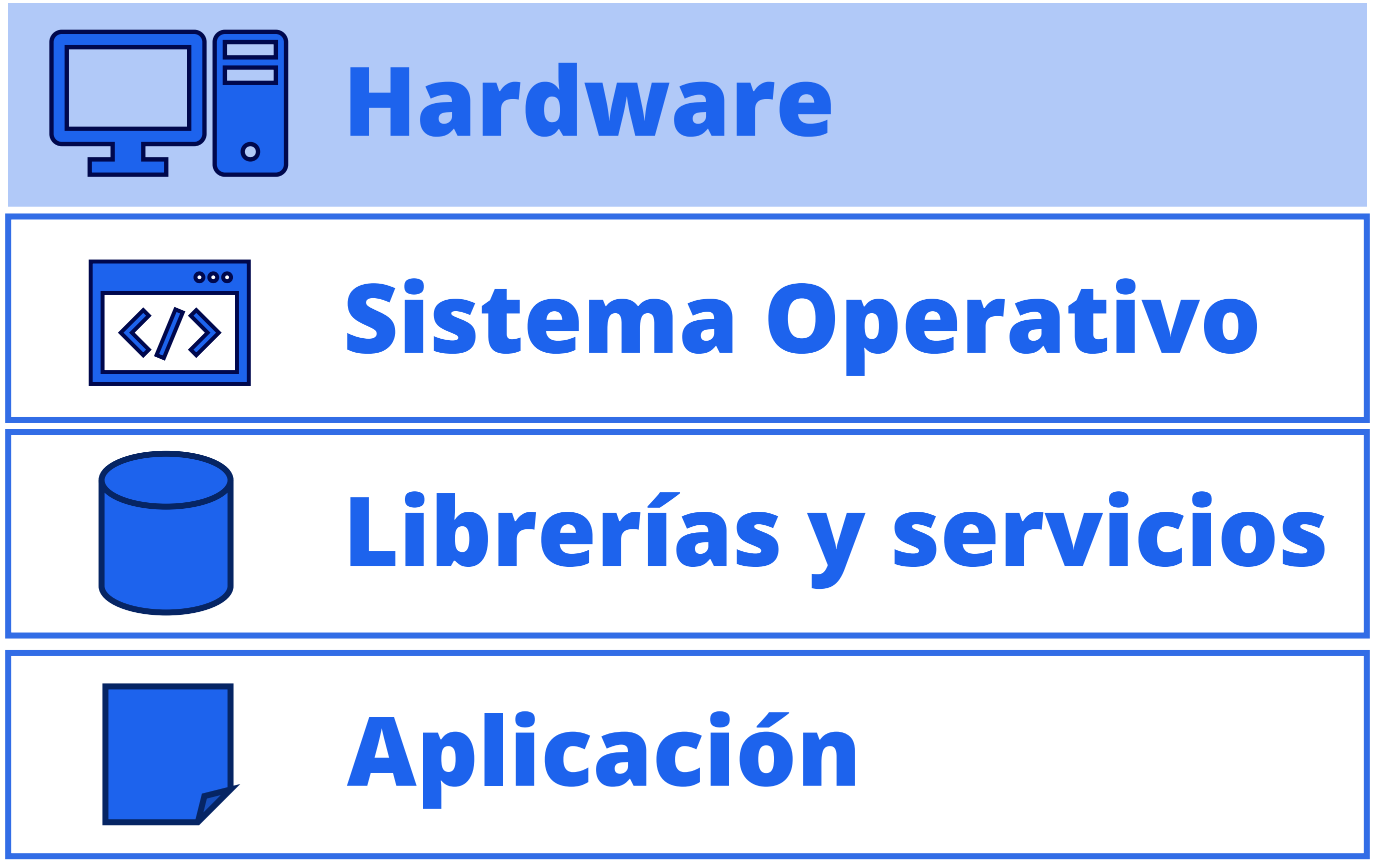
* Ambos incluidos en la descarga de Docker Desktop.

Conocimientos

- Básico de redes, shell y procesos

Una aplicación no es solo el código

Las dependencias



Hardware

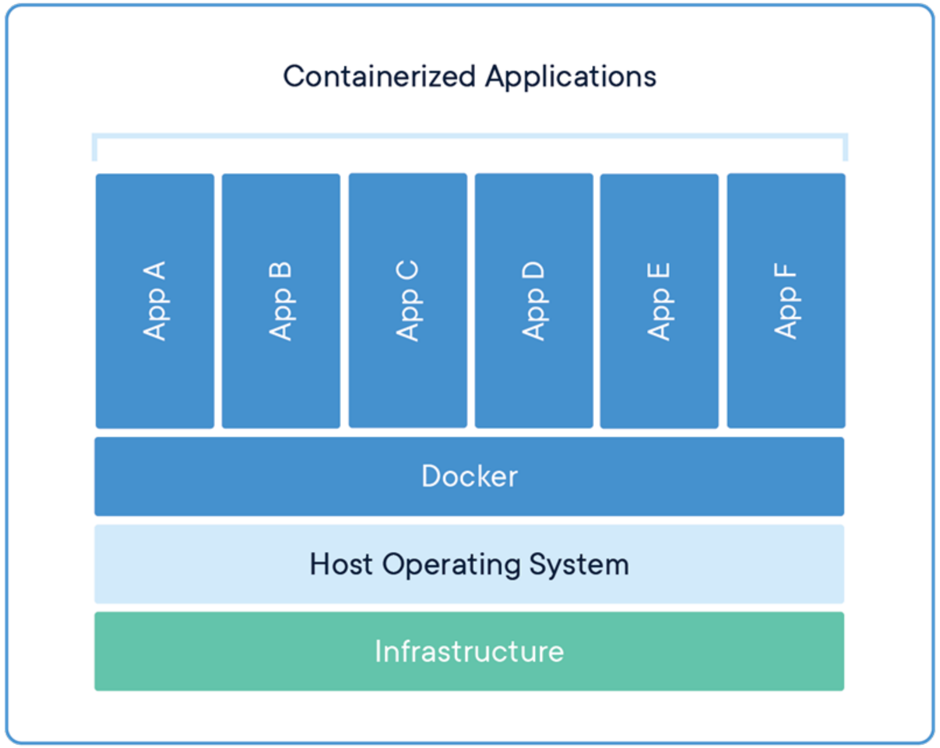
Sistema Operativo

Librerías y servicios

Aplicación

¿Qué es un contenedor?

Docker y los contenedores

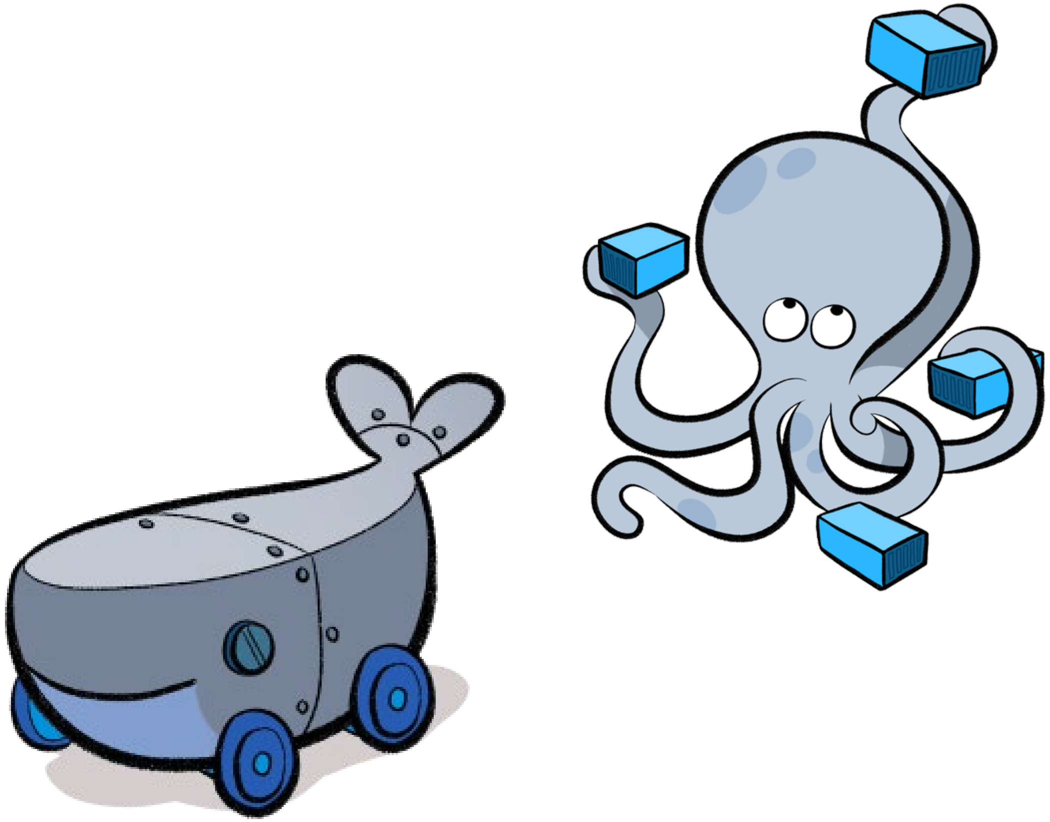


Containerized Applications

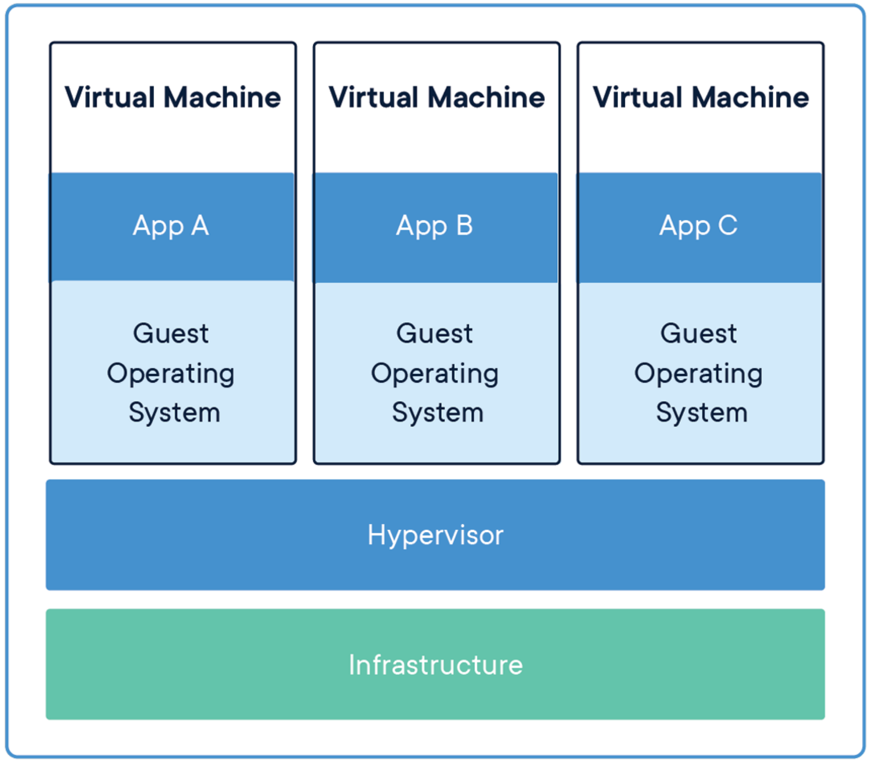
Docker como plataforma

Docker Desktop y plugins

- Docker Engine
- Docker CLI client
- Docker Scout
- Docker Buildx
- Docker Extensions
- Docker Compose
- Kubernetes

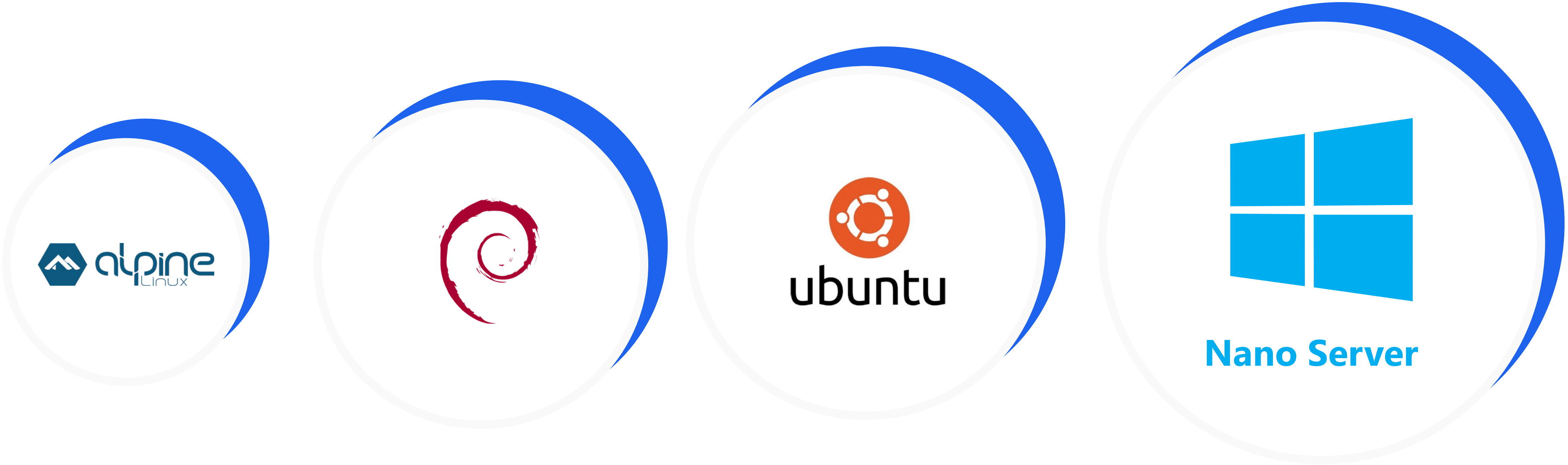


¿Máquinas virtuales?



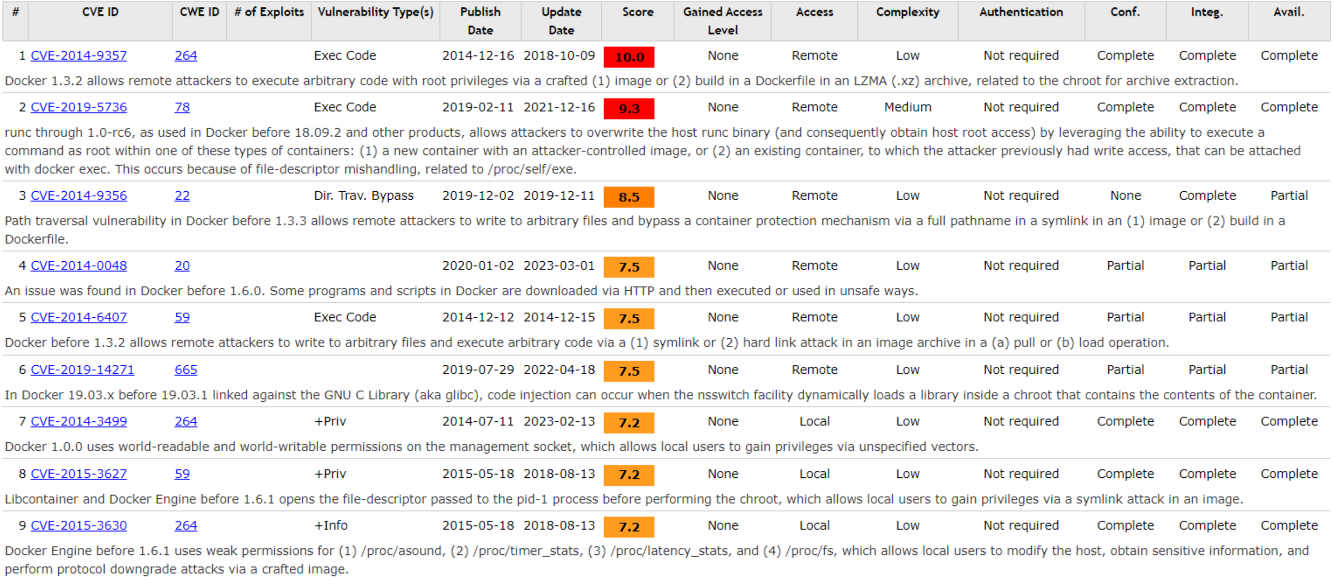
Sistemas Operativos

“El tamaño importa”



Seguridad

“Nada es 100% seguro”



<table>
<tr><th>#</th><th>CVE ID</th><th>CWE ID</th><th># of Exploits</th><th>Vulnerability Type(s)</th><th>Publish Date</th><th>Update Date</th><th>Score</th><th>Gained Access Level</th><th>Access</th><th>Complexity</th><th>Authentication</th><th>Conf.</th><th>Integ.</th><th>Avail.</th></tr>
<tr><td>1</td><td>CVE-2014-9357</td><td>264</td><td></td><td>Exec Code</td><td>2014-12-16</td><td>2018-10-09</td><td>10.0</td><td>None</td><td>Remote</td><td>Low</td><td>Not required</td><td>Complete</td><td>Complete</td><td>Complete</td></tr>
<tr><td colspan="15">Docker 1.3.2 allows remote attackers to execute arbitrary code with root privileges via a crafted (1) image or (2) build in a Dockerfile in an LZMA (.xz) archive, related to the chroot for archive extraction.</td></tr>
<tr><td>2</td><td>CVE-2019-5736</td><td>78</td><td></td><td>Exec Code</td><td>2019-02-11</td><td>2021-12-16</td><td>9.3</td><td>None</td><td>Remote</td><td>Medium</td><td>Not required</td><td>Complete</td><td>Complete</td><td>Complete</td></tr>
<tr><td colspan="15">runc through 1.0-rc6, as used in Docker before 18.09.2 and other products, allows attackers to overwrite the host runc binary (and consequently obtain host root access) by leveraging the ability to execute a command as root within one of these types of containers: (1) a new container with an attacker-controlled image, or (2) an existing container, to which the attacker previously had write access, that can be attached with docker exec. This occurs because of file-descriptor mishandling, related to /proc/self/exe.</td></tr>
<tr><td>3</td><td>CVE-2014-9356</td><td>22</td><td></td><td>Dir. Trav. Bypass</td><td>2019-12-02</td><td>2019-12-11</td><td>8.5</td><td>None</td><td>Remote</td><td>Low</td><td>Not required</td><td>None</td><td>Complete</td><td>Partial</td></tr>
<tr><td colspan="15">Path traversal vulnerability in Docker before 1.3.3 allows remote attackers to write to arbitrary files and bypass a container protection mechanism via a full pathname in a symlink in an (1) image or (2) build in a Dockerfile.</td></tr>
<tr><td>4</td><td>CVE-2014-0048</td><td>20</td><td></td><td></td><td>2020-01-02</td><td>2023-03-01</td><td>7.5</td><td>None</td><td>Remote</td><td>Low</td><td>Not required</td><td>Partial</td><td>Partial</td><td>Partial</td></tr>
<tr><td colspan="15">An issue was found in Docker before 1.6.0. Some programs and scripts in Docker are downloaded via HTTP and then executed or used in unsafe ways.</td></tr>
<tr><td>5</td><td>CVE-2014-6407</td><td>59</td><td></td><td>Exec Code</td><td>2014-12-12</td><td>2014-12-15</td><td>7.5</td><td>None</td><td>Remote</td><td>Low</td><td>Not required</td><td>Partial</td><td>Partial</td><td>Partial</td></tr>
<tr><td colspan="15">Docker before 1.3.2 allows remote attackers to write to arbitrary files and execute arbitrary code via a (1) symlink or (2) hard link attack in an image archive in a (a) pull or (b) load operation.</td></tr>
<tr><td>6</td><td>CVE-2019-14271</td><td>665</td><td></td><td></td><td>2019-07-29</td><td>2022-04-18</td><td>7.5</td><td>None</td><td>Remote</td><td>Low</td><td>Not required</td><td>Partial</td><td>Partial</td><td>Partial</td></tr>
<tr><td colspan="15">In Docker 19.03.x before 19.03.1 linked against the GNU C Library (aka glibc), code injection can occur when the nsswitch facility dynamically loads a library inside a chroot that contains the contents of the container.</td></tr>
<tr><td>7</td><td>CVE-2014-3499</td><td>264</td><td></td><td>+Priv</td><td>2014-07-11</td><td>2023-02-13</td><td>7.2</td><td>None</td><td>Local</td><td>Low</td><td>Not required</td><td>Complete</td><td>Complete</td><td>Complete</td></tr>
<tr><td colspan="15">Docker 1.0.0 uses world-readable and world-writable permissions on the management socket, which allows local users to gain privileges via unspecified vectors.</td></tr>
<tr><td>8</td><td>CVE-2015-3627</td><td>59</td><td></td><td>+Priv</td><td>2015-05-18</td><td>2018-08-13</td><td>7.2</td><td>None</td><td>Local</td><td>Low</td><td>Not required</td><td>Complete</td><td>Complete</td><td>Complete</td></tr>
<tr><td colspan="15">Libcontainer and Docker Engine before 1.6.1 opens the file-descriptor passed to the pid-1 process before performing the chroot, which allows local users to gain privileges via a symlink attack in an image.</td></tr>
<tr><td>9</td><td>CVE-2015-3630</td><td>264</td><td></td><td>+Info</td><td>2015-05-18</td><td>2018-08-13</td><td>7.2</td><td>None</td><td>Local</td><td>Low</td><td>Not required</td><td>Complete</td><td>Complete</td><td>Complete</td></tr>
<tr><td colspan="15">Docker Engine before 1.6.1 uses weak permissions for (1) /proc/asound, (2) /proc/timer_stats, (3) /proc/latency_stats, and (4) /proc/fs, which allows local users to modify the host, obtain sensitive information, and perform protocol downgrade attacks via a crafted image.</td></tr>
</table>

Docker Daemon

Servidor

dockerd

Cliente (CLI)

docker ...

Conceptos (I)

Básico

Dockerfile

Imagen

Contenedor

Conceptos (II)

Programación

Dockerfile

Imagen

Contenedor

Dockerfile (I)

Notación

```
# Comentario
INSTRUCCIÓN argumentos
```

Dockerfile (II)

Contenerizando tu aplicación

FROM imagen[:versión]

Partir de una imagen

RUN comando

Actualizar la imagen

COPY archivo_host directorio_dentro

Añadir archivos a tu imagen

ADD archivo/enlace directorio_dentro

Añadir archivos a tu imagen

Dockerfile (III)

Configurando tu contenedor

ENV variable_de_entorno

Añadir variables de entorno para la construcción de la imagen y contenedor

ARG argumento

Tomar argumentos/variables para la construcción de la imagen

Dockerfile (IV)

Lanzando tu contenedor

CMD [“comando”, “param “ . . .]

CMD comando param . . .

CMD param1 param2 . . .

Comando (y/o parámetros) que el contenedor ejecuta al iniciar

ENTRYPOINT comando param . . .

ENTRYPOINT [“comando”, “param “ . . .]

Comando que el contenedor ejecuta al iniciar

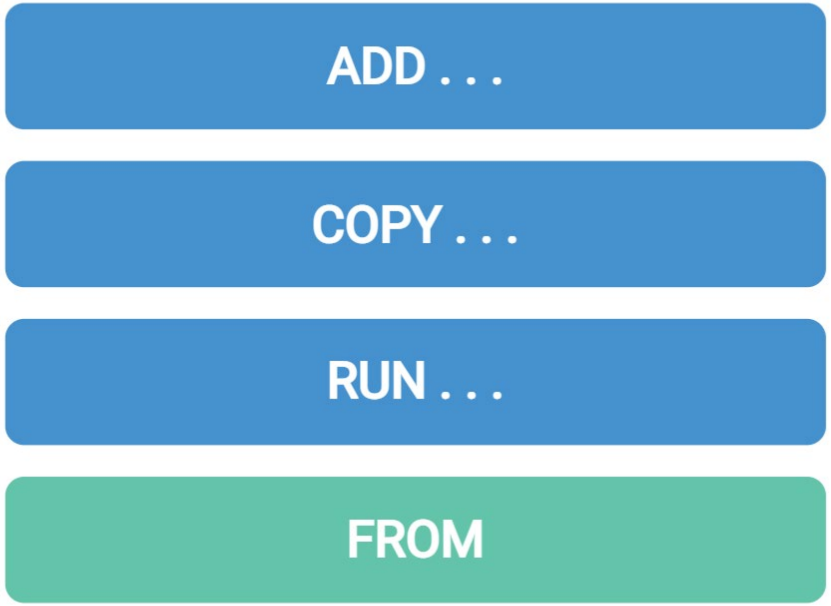
Dockerfile (V)

Ejemplo simple

```
FROM alpine:latest
COPY ./script.sh .
CMD ./script.sh
```

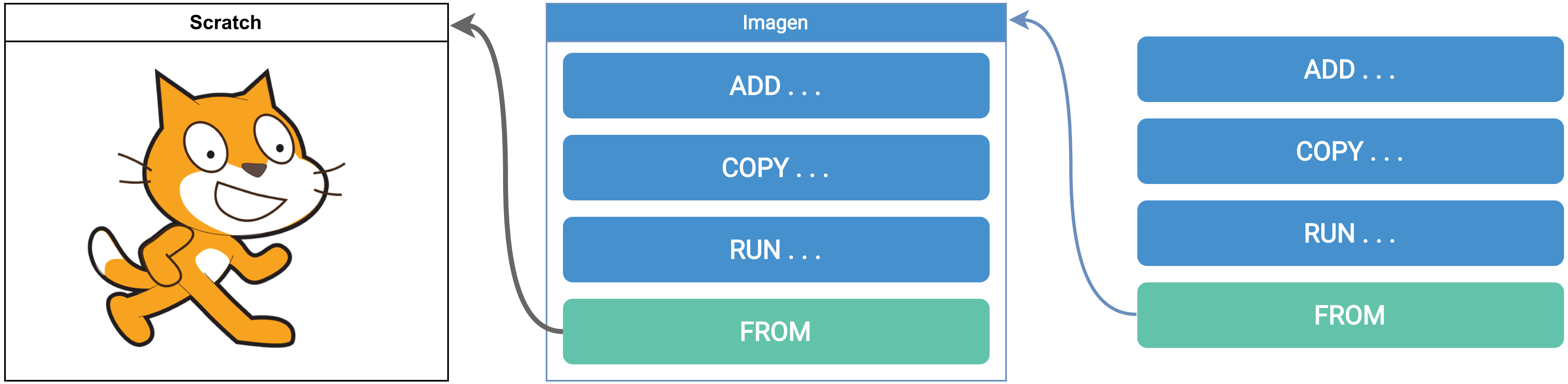
Imágenes (I)

Guardando nuestro trabajo



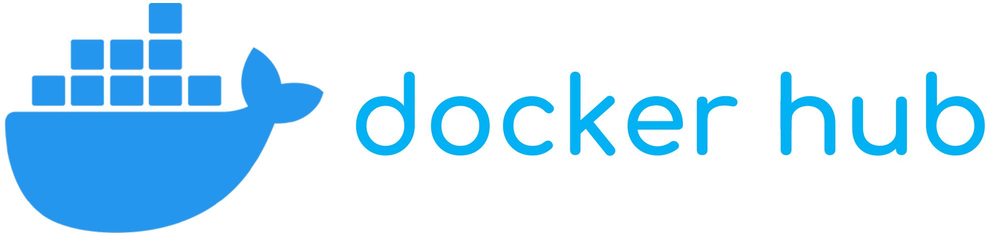
Imágenes (II)

Desde los orígenes



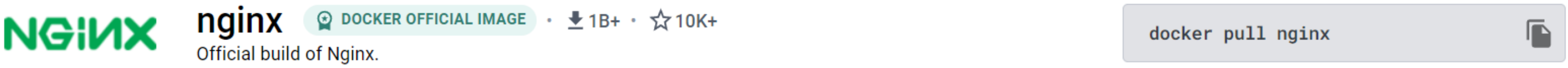
Docker Registry

Dockerhub, me suena...



Dockerhub

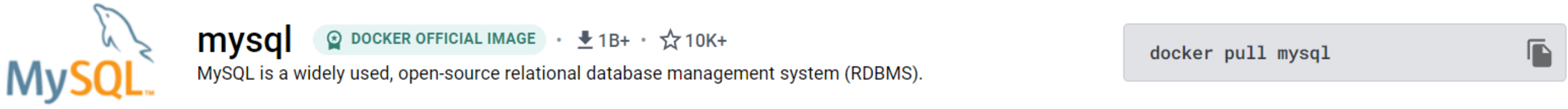
No hagas todo el trabajo



nginx DOCKER OFFICIAL IMAGE · ⭳1B+ · ☆10K+

Official build of Nginx.

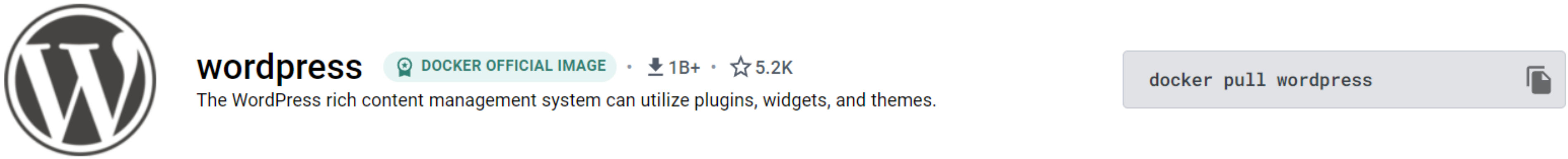
```
docker pull nginx
```



mysql DOCKER OFFICIAL IMAGE · ⭳1B+ · ☆10K+

MySQL is a widely used, open-source relational database management system (RDBMS).

```
docker pull mysql
```



wordpress DOCKER OFFICIAL IMAGE · ⭳1B+ · ☆5.2K

The WordPress rich content management system can utilize plugins, widgets, and themes.

```
docker pull wordpress
```

Comandos (I)

Imágenes

```
docker image build [-f archivo] directorio
docker image ls
```

Dockerhub

```
docker push imagen
docker pull imagen
docker tag tag_fuente tag_destino
```

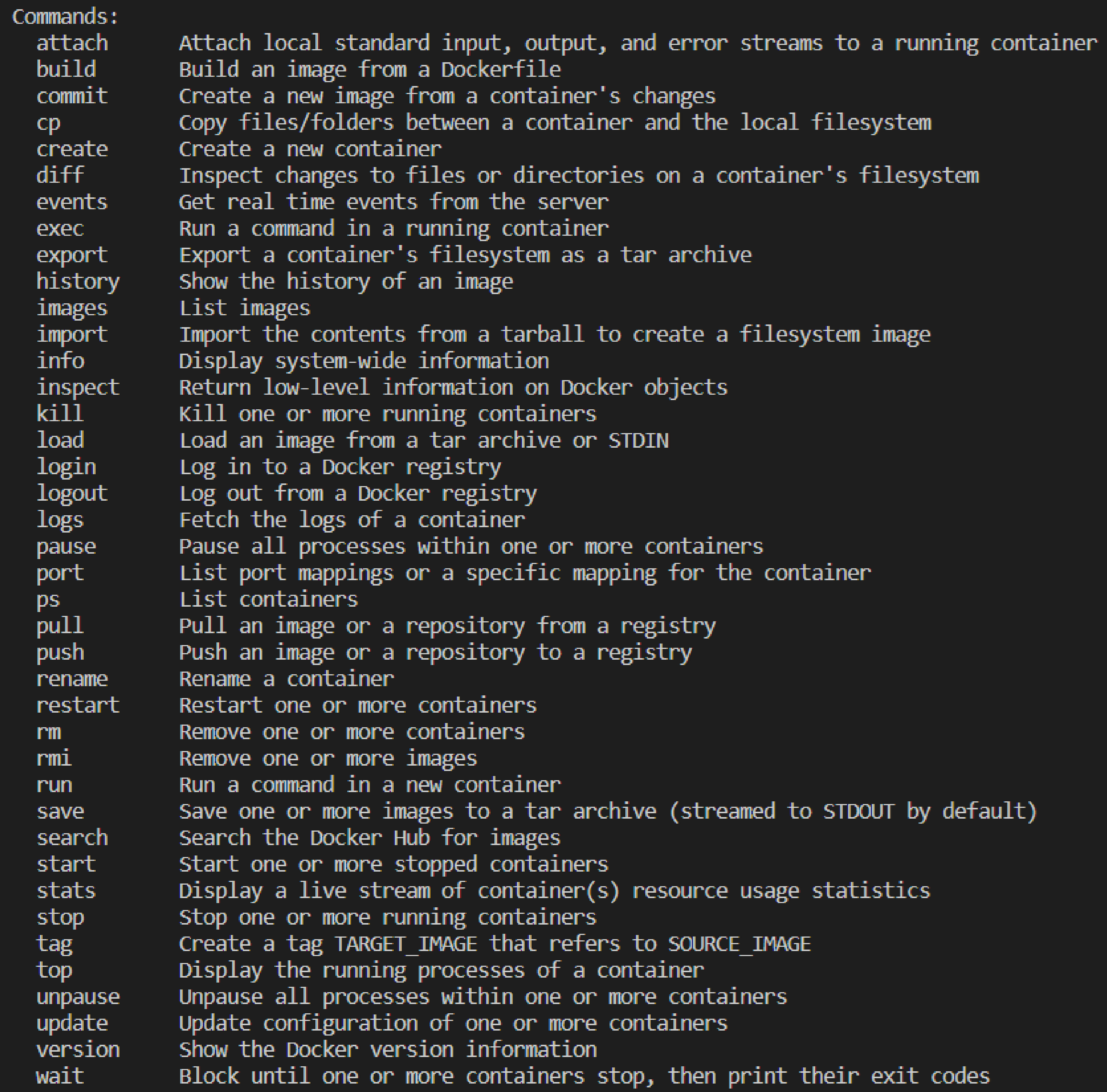
Comandos (II)

Contenedores

```
docker container run  [opciones] imagen
docker container start contenedor
docker container stop contenedor
docker container ls
docker container prune
```

Opciones de interés : -d –rm –it –p –v –e . . .

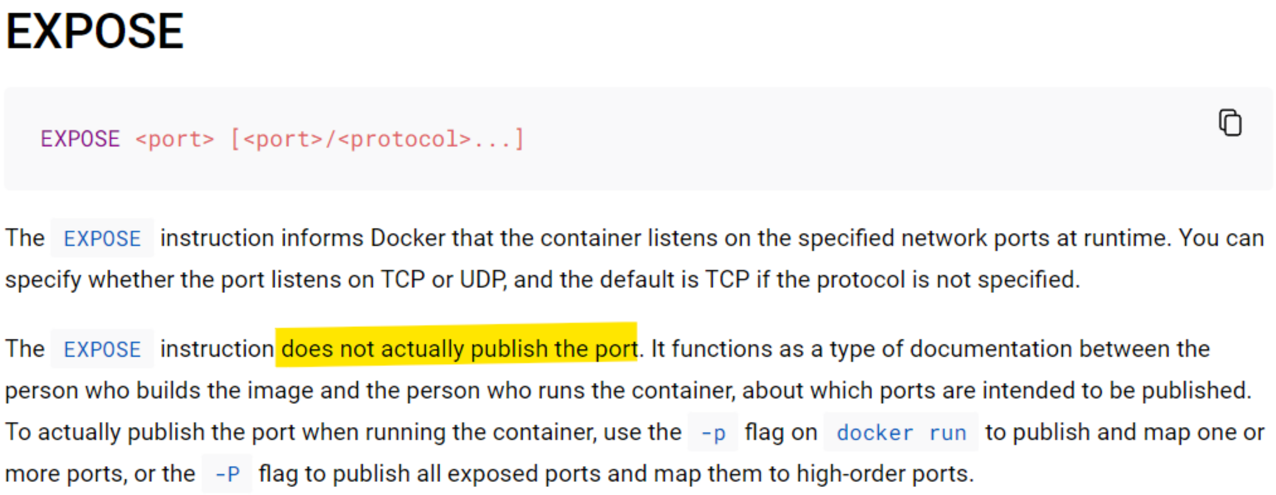
Comandos de Docker



```
Commands:
  attach      Attach local standard input, output, and error streams to a running container
  build       Build an image from a Dockerfile
  commit      Create a new image from a container's changes
  cp          Copy files/folders between a container and the local filesystem
  create      Create a new container
  diff        Inspect changes to files or directories on a container's filesystem
  events      Get real time events from the server
  exec        Run a command in a running container
  export      Export a container's filesystem as a tar archive
  history     Show the history of an image
  images      List images
  import      Import the contents from a tarball to create a filesystem image
  info        Display system-wide information
  inspect     Return low-level information on Docker objects
  kill        Kill one or more running containers
  load        Load an image from a tar archive or STDIN
  login       Log in to a Docker registry
  logout      Log out from a Docker registry
  logs        Fetch the logs of a container
  pause       Pause all processes within one or more containers
  port        List port mappings or a specific mapping for the container
  ps          List containers
  pull        Pull an image or a repository from a registry
  push        Push an image or a repository to a registry
  rename      Rename a container
  restart     Restart one or more containers
  rm          Remove one or more containers
  rmi         Remove one or more images
  run         Run a command in a new container
  save        Save one or more images to a tar archive (streamed to STDOUT by default)
  search      Search the Docker Hub for images
  start       Start one or more stopped containers
  stats       Display a live stream of container(s) resource usage statistics
  stop        Stop one or more running containers
  tag         Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE
  top         Display the running processes of a container
  unpause     Unpause all processes within one or more containers
  update      Update configuration of one or more containers
  version     Show the Docker version information
  wait        Block until one or more containers stop, then print their exit codes
```

Documentación (I)

Leer atentamente



EXPOSE

```
EXPOSE <port> [<port>/<protocol>...]
```

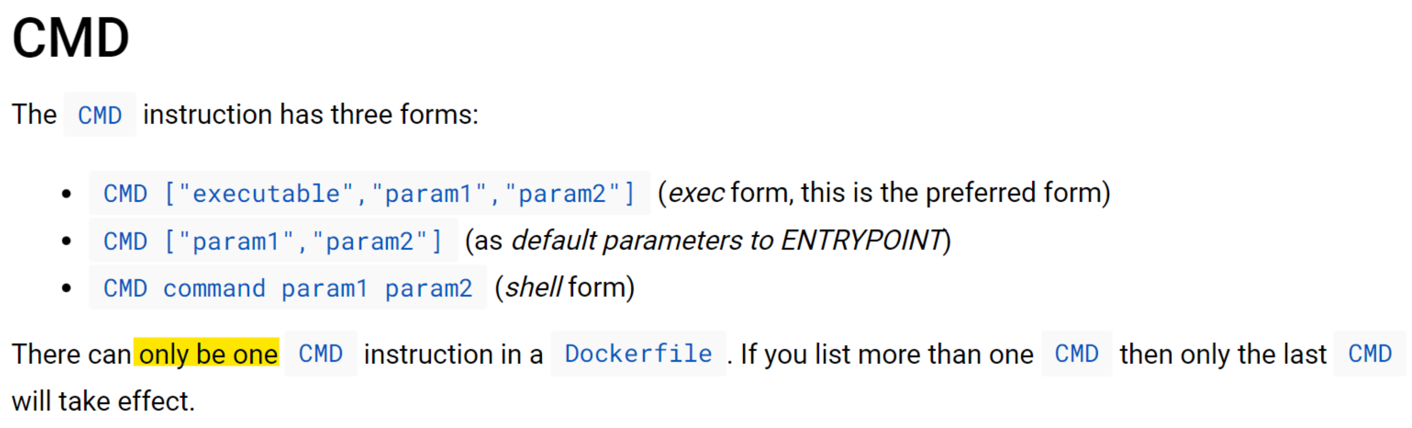
The `EXPOSE` instruction informs Docker that the container listens on the specified network ports at runtime. You can specify whether the port listens on TCP or UDP, and the default is TCP if the protocol is not specified.

The `EXPOSE` instruction does not actually publish the port. It functions as a type of documentation between the person who builds the image and the person who runs the container, about which ports are intended to be published. To actually publish the port when running the container, use the `-p` flag on `docker run` to publish and map one or more ports, or the `-P` flag to publish all exposed ports and map them to high-order ports.

[Documentación de Docker (docker.docs)]()

Documentación (II)

CMD



CMD

The `CMD` instruction has three forms:

- `CMD ["executable","param1","param2"]` (*exec* form, this is the preferred form)
- `CMD ["param1","param2"]` (as *default parameters to ENTRYPOINT*)
- `CMD command param1 param2` (*shell* form)

There can only be one `CMD` instruction in a `Dockerfile`. If you list more than one `CMD` then only the last `CMD` will take effect.

[Documentación de Docker (docker.docs)](#)

A Practicar

Ejercicios

Recomendaciones

1. Pregunta a tus compañeros antes que a una IA generativa.

2. Usa Docker Desktop para iniciar el demonio, pero no para realizar los ejercicios.

Ejercicio 0

Dummy Dockerfile

josesanc02/taller-00

Partiendo de la imagen, añadir un archivo ‘dummy’

Comandos (Unix):

- **touch (crear ficheros)**

Soluciones (I)

El primer Dockerfile

0. Dockerfile

```
FROM josesanc02/taller-00
RUN touch dummy
```

0. Comandos

```
docker build –t etiqueta .
docker run etiqueta
```

Comandos - Ejercicio 1

Echa a correr

josesanc02/taller-01

Descarga la imagen y descubre qué se esconde en localhost (http://127.0.0.1)

Comandos - Ejercicio 2

El sentido de la vida, el universo y todo lo demás

josesanc02/taller-02

Comandos - Ejercicio 3

Un secreto mal guardado

josesanc02/taller-03

Comandos (Unix):

- /bin/sh
- cat (leer ficheros)
- ls (listar directorio)

Soluciones (II)

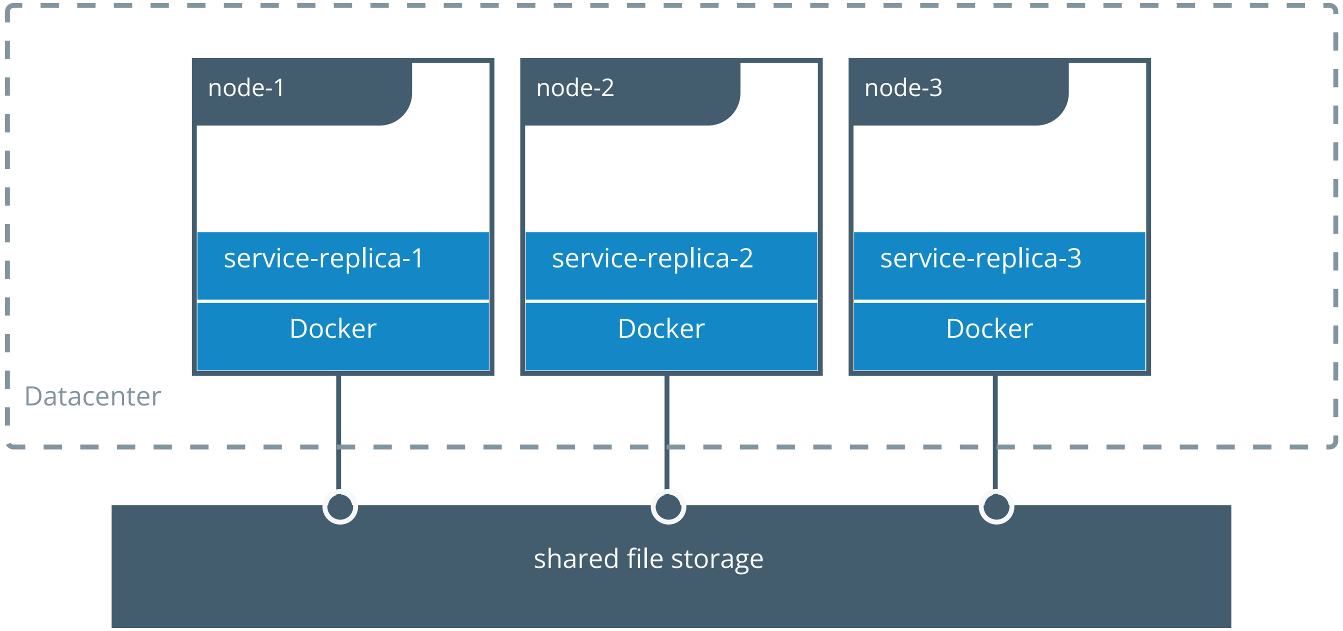
Agora sim entendo

1. docker run –p 8080:80 imagen
2. docker run –e THEANSWERTOLIFE=42 imagen
3. docker run –it imagen /bin/sh

Mecanismos Entre Contenedores

Volúmenes (I)

La persistencia



Volúmenes (II)

Volúmenes de contenedor

```
docker volume create name

docker run ... -v <name>:<ruta_ contenedor>
```

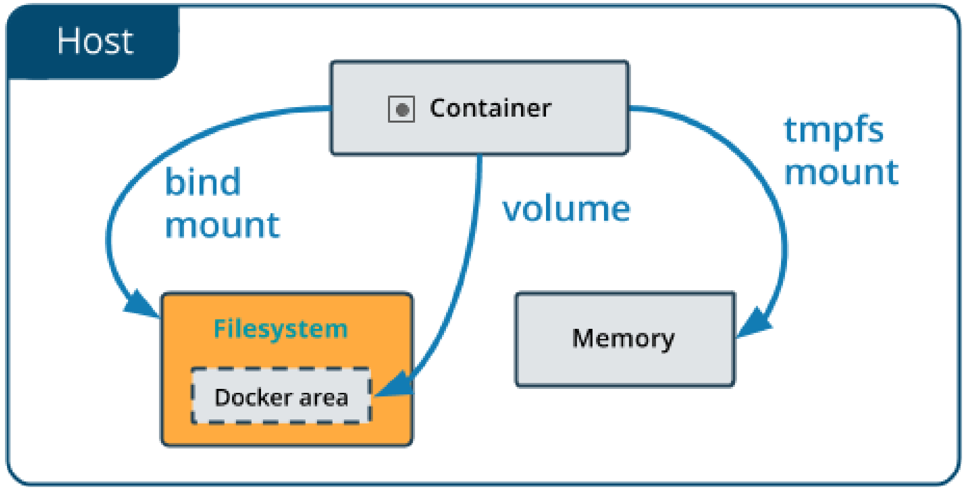
¿Volúmenes de directorio?

```
<ruta_host>:<ruta_contenedor>
```

Bind mounts

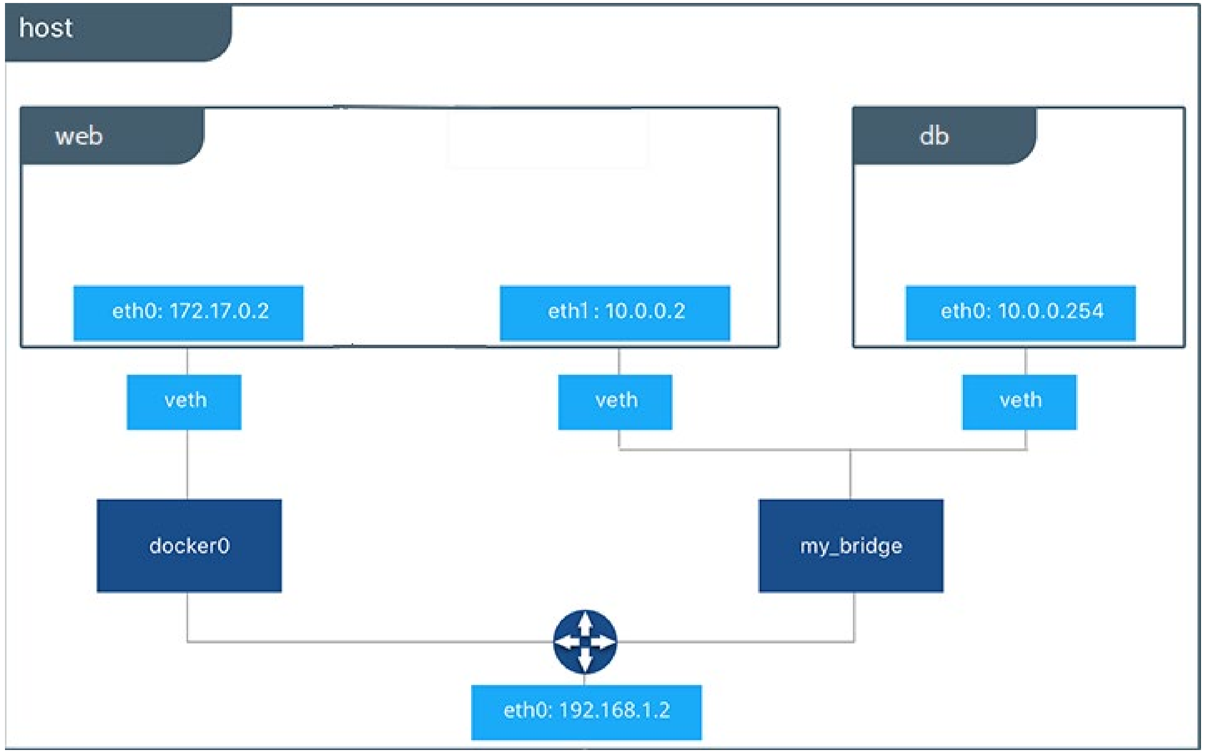
Compartiendo el sistema

```
--mount
target="<ruta_host>",
source="<ruta_contenedor>"
```



Networks (I)

10.X.Y.Z...



Docker Compose

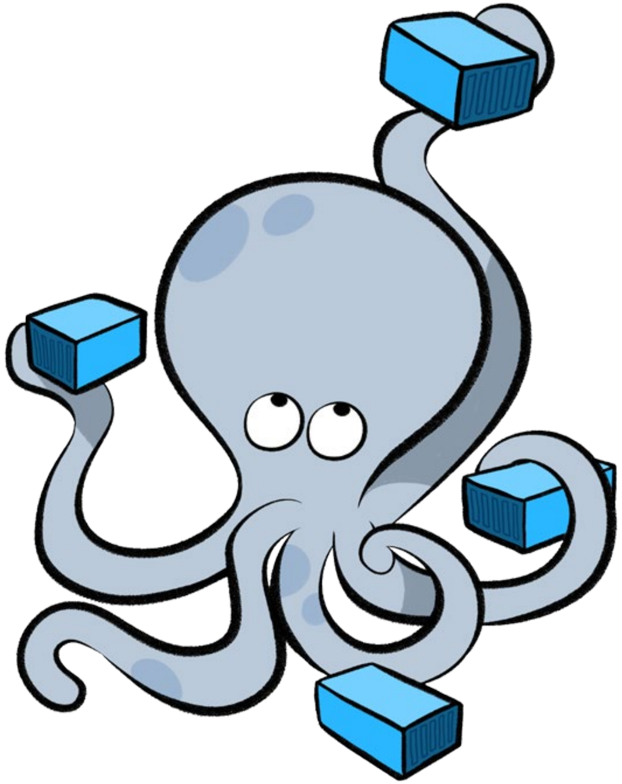
Docker Compose (I)

Dando un poco de orden

Services
Los servicios/contenedores que se echan a correr.

Volumes
Dónde guardar la información.

Networks
Cómo conectarlos los contenedores.



Docker Compose (II)

Comandos

[v.1] docker-compose subcomando -- DEPRECATED

[v.2+] docker compose subcomando

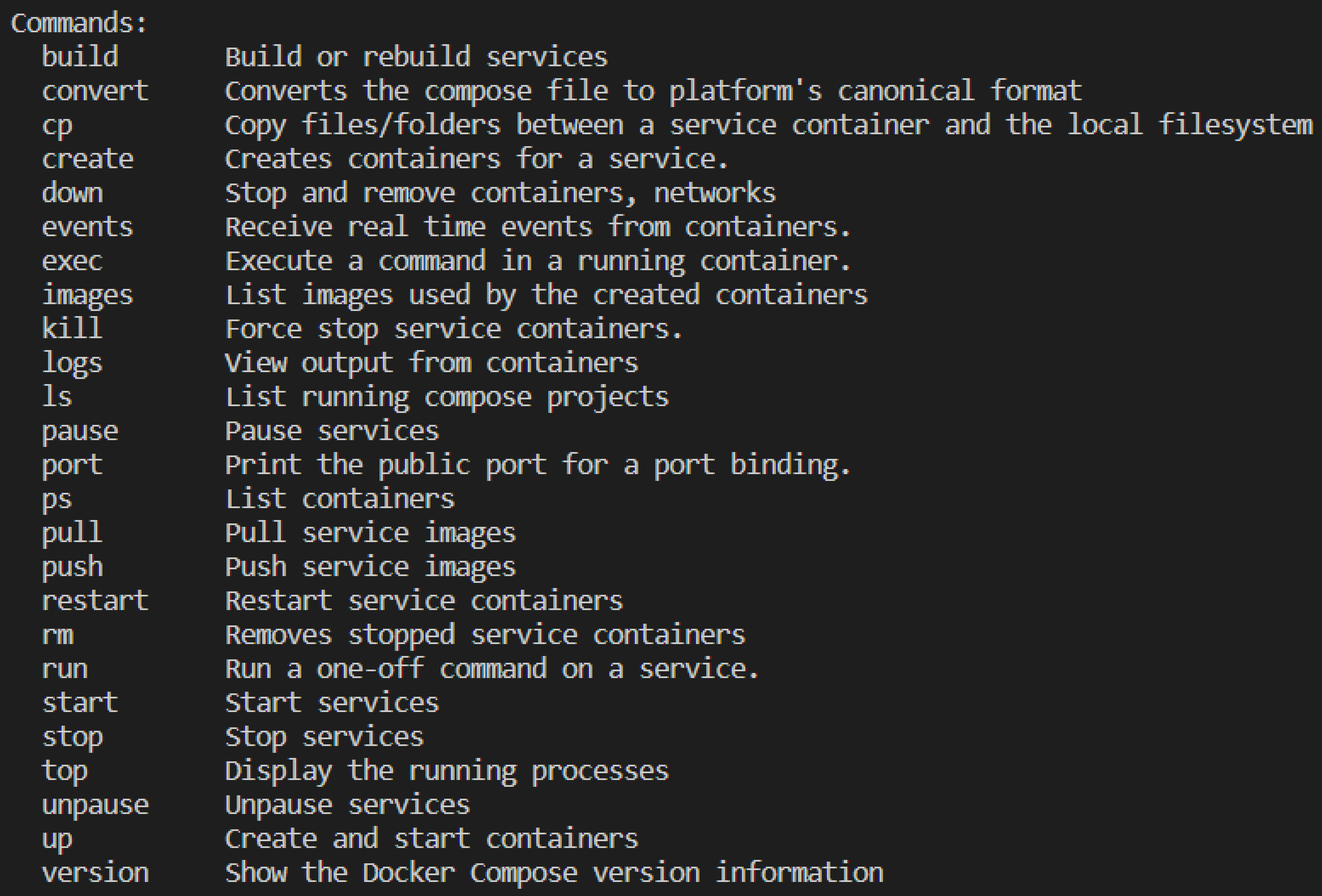
docker compose up

Iniciar servicios del docker-compose.yml del directorio actual

docker compose down

Para y borra los servicios del comando docker compose up

Comandos de Docker Compose



```
Commands:
  build       Build or rebuild services
  convert     Converts the compose file to platform's canonical format
  cp          Copy files/folders between a service container and the local filesystem
  create      Creates containers for a service.
  down        Stop and remove containers, networks
  events      Receive real time events from containers.
  exec        Execute a command in a running container.
  images      List images used by the created containers
  kill        Force stop service containers.
  logs        View output from containers
  ls          List running compose projects
  pause       Pause services
  port        Print the public port for a port binding.
  ps          List containers
  pull        Pull service images
  push        Push service images
  restart     Restart service containers
  rm          Removes stopped service containers
  run         Run a one-off command on a service.
  start       Start services
  stop        Stop services
  top         Display the running processes
  unpause     Unpause services
  up          Create and start containers
  version     Show the Docker Compose version information
```

Compose File (v.3) - I

La estructura

```
version: ‘versión'

services:
  nombre_de_servicio:


networks:
  nombre_de_red:
   . . .
volumes:
  nombre_de_volumen:
     . . .
 y más . . .
```

Compose File (v.3) - II

Configuración en docker-compose.yml

```
  nombre_servicio_1:
    container_name: nombre_contenedor
    image: nombre_para_la_imagen
    build:
     context: ruta
     dockerfile: archivo_dockerfile
     args:
       - clave=valor
    environment:
       - clave=valor
    ports:
      - “8000:80”
```

Compose File (v.3) - III

Más atributos. . .

```
  nombre_servicio_2:
    image: imagen_de_registry
    restart: on-failure
    env_file: archivo.env
    depends_on:
      - nombre_servicio_1
    expose:
      - 8000
```

Y muchos más (Compose file version 3 reference)

Compose File (v.3) - IV

Configurando las conexiones

```
(services:)
  nombre_servicio_2:
    ...
    networks:
      nombre_de_red
    volumes:
      - ruta_host:ruta_contenedor
      - nombre_de_volumen:ruta_contenedor

networks:
  nombre_de_red:
    ...
volumes:
  nombre_de_volumen:
      ...
```

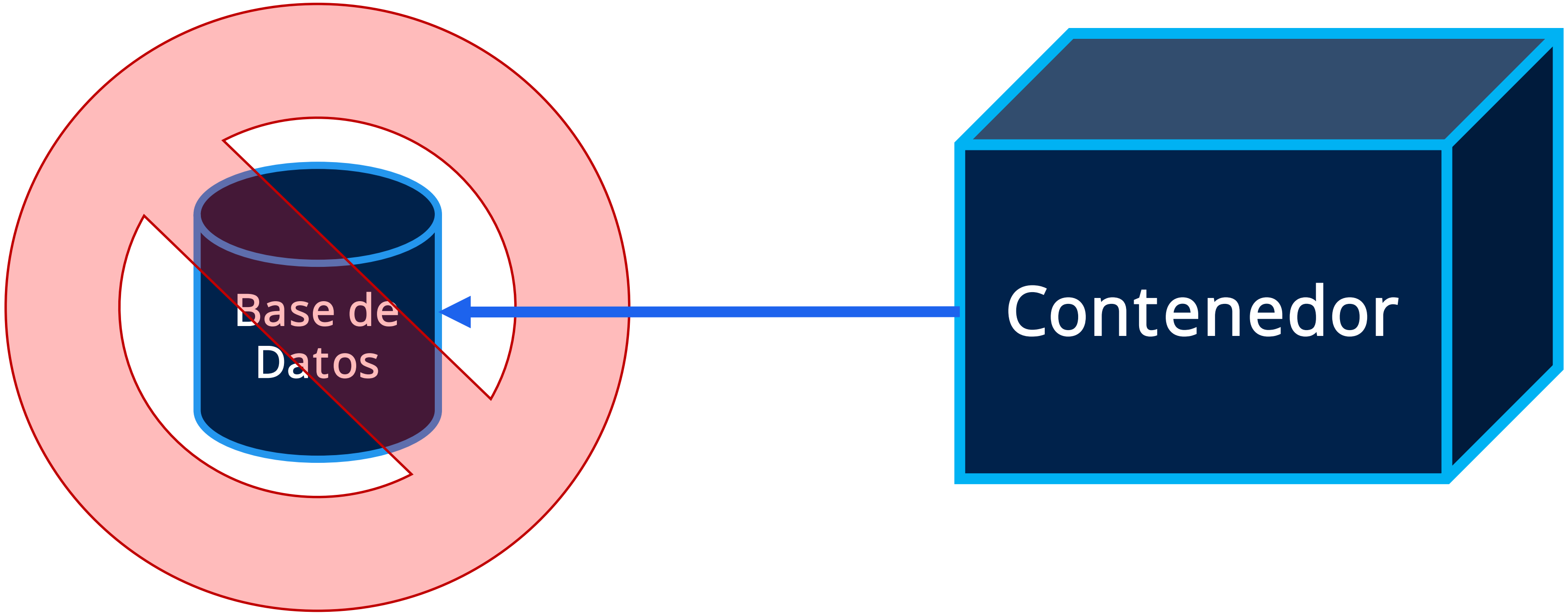
Compose File (v.3) - V

Comprobando los errores

docker compose config

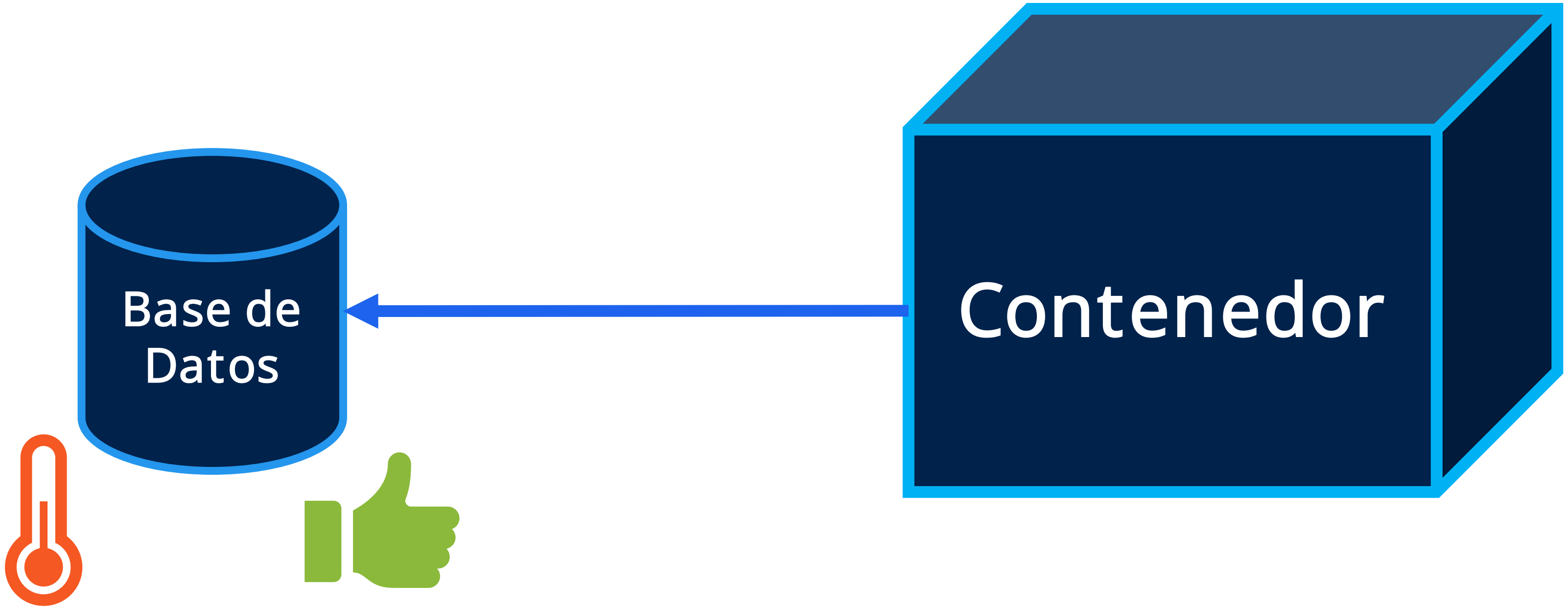
Organizando dependencias

depends_on



Comprobando dependencias

Healthcheck y service_healthy



.env

Guardando variables de entorno

```
DB_HOST=ejemplo.com
DB_PORT=5432
```

```
DB_USER=user
DB_PASSWORD=password
```

Usando variables de entorno

```
$DB_HOST
```

```
${DB_PASSWORD}
```

Compose Ejemplo

Wordpress + MySQL

Configura un docker-compose.yml con wordpress y mysql

Visita la página de Wordpress de Dockerhub

Ejercicio 4

¿Dónde guardo mis datos?

josesanc02/taller-04

La aplicación ya está hecha, pero dónde guardo mis datos...

CURIOSIDADES

¿Dockerfile y compose.yml automático?

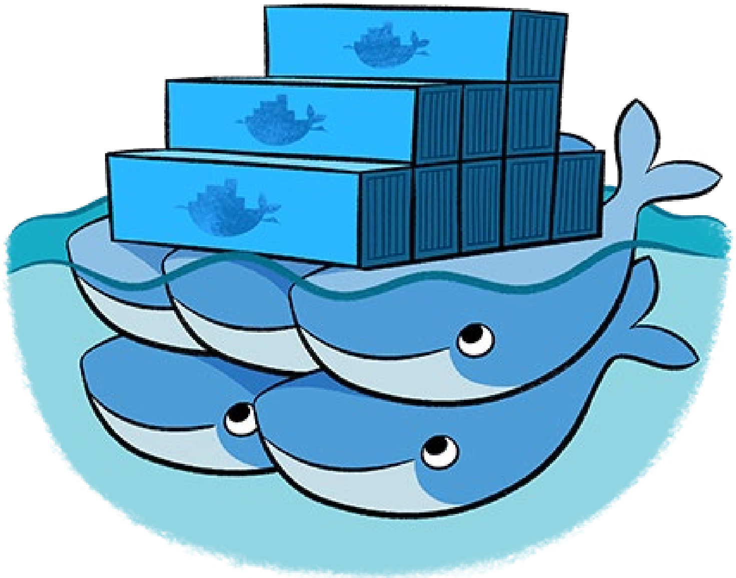
Rápido y con buenas prácticas

docker init

Dockerfile
compose.yml
.dockerignore

Orquestradores

Kubernetes



Docker Swarm

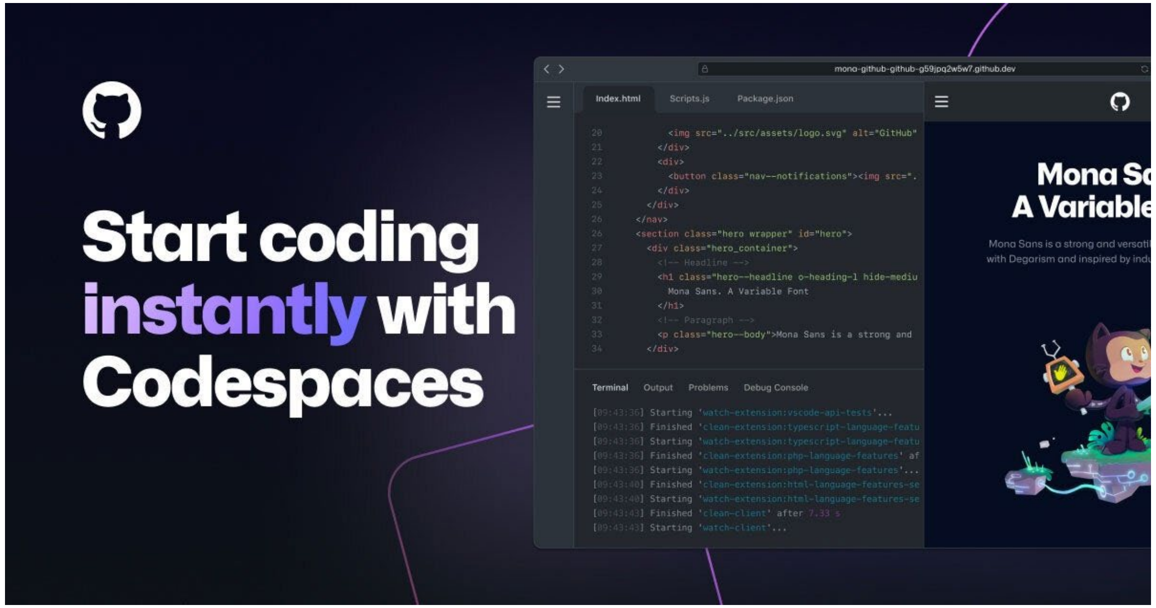
Podman

alias docker=podman

- **Compatible con Kubernetes**
- **Por Red Hat**

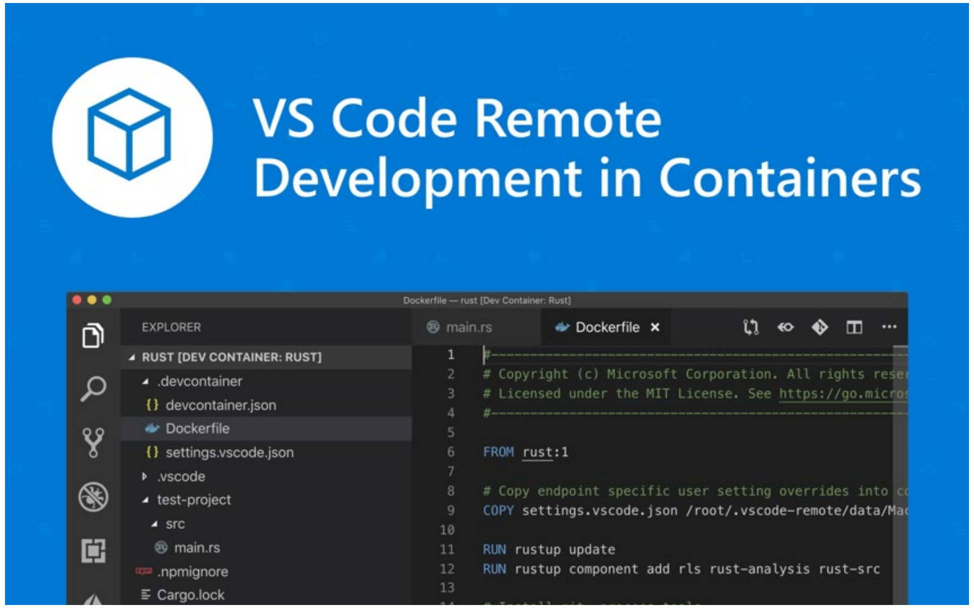
Desarrollando en contenedores

Devcontainers


Start coding instantly with Codespaces
mono-github-github-g59jpq2w5w7.github.dev
Index.html
Scripts.js
Package.json
Terminal
Output
Problems
Debug Console
Mona Sa
A Variable
Mona Sans is a strong and versati
with Degarism and inspired by indu

```
            <img src="../src/assets/logo.svg" alt="GitHub"
          </div>
          <div>
            <button class="nav--notifications"><img src=".
          </div>
        </div>
      </nav>
      <section class="hero wrapper" id="hero">
        <div class="hero_container">
          <!-- Headline -->
          <h1 class="hero--headline o-heading-l hide-mediu
            Mona Sans. A Variable Font
          </h1>
          <!-- Paragraph -->
          <p class="hero--body">Mona Sans is a strong and
        </div>
```

```
[09:43:36] Starting 'watch-extension:vscode-api-tests'...
[09:43:36] Finished 'clean-extension:typescript-language-featu
[09:43:36] Starting 'watch-extension:typescript-language-featu
[09:43:36] Finished 'clean-extension:php-language-features' af
[09:43:36] Starting 'watch-extension:php-language-features'...
[09:43:40] Finished 'clean-extension:html-language-features-se
[09:43:40] Starting 'watch-extension:html-language-features-se
[09:43:43] Finished 'clean-client' after 7.33 s
[09:43:43] Starting 'watch-client'...
```

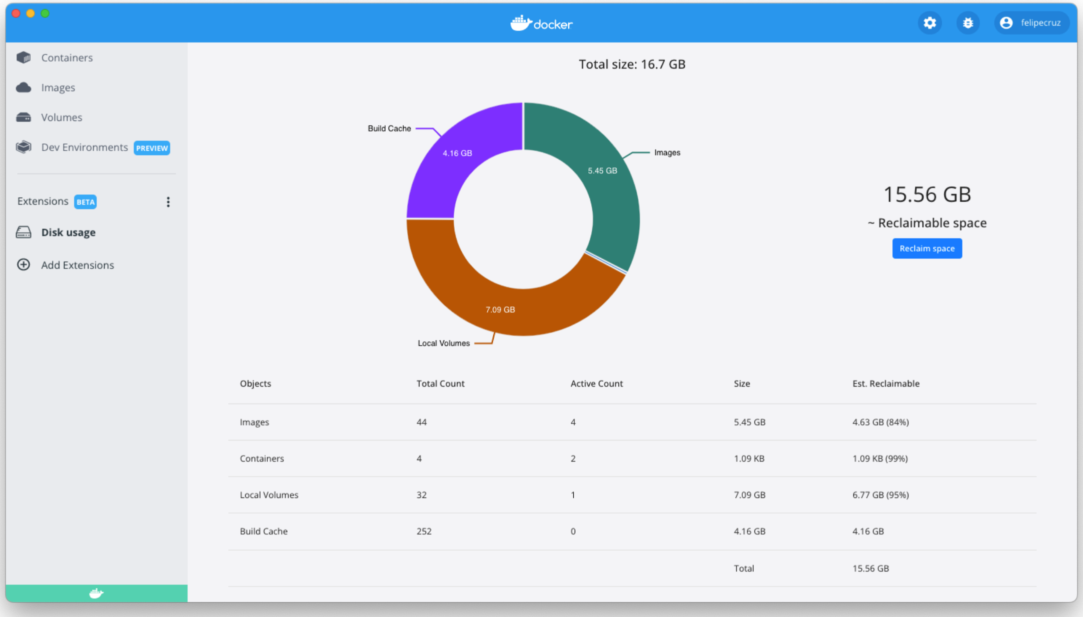


```
#-----------------------------------------------------------------
# Copyright (c) Microsoft Corporation. All rights reser
# Licensed under the MIT License. See https://go.micros
#-----------------------------------------------------------------

FROM rust:1

# Copy endpoint specific user setting overrides into co
COPY settings.vscode.json /root/.vscode-remote/data/Mac

RUN rustup update
RUN rustup component add rls rust-analysis rust-src

```

Otros consejos

Haciendo limpieza, prune

- **Containers**
- **Images**
- **Volumes**



Total size: 16.7 GB

15.56 GB

~ Reclaimable space

Objects	Total Count	Active Count	Size	Est. Reclaimable
Images	44	4	5.45 GB	4.63 GB (84%)
Containers	4	2	1.09 KB	1.09 KB (99%)
Local Volumes	32	1	7.09 GB	6.77 GB (95%)
Build Cache	252	0	4.16 GB	4.16 GB
			Total	15.56 GB

Errores comunes

404 – Not found

failed to solve with frontend dockerfile.v0: failed to read dockerfile

No se encuentra el Dockerfile, el nombre es incorrecto o no estás en el directorio indicado.

Fallos de **identación** en el archivo .yml

Nombre del servicio incorrecto (**DNS**)

Puertos sin **configurar**/exponer

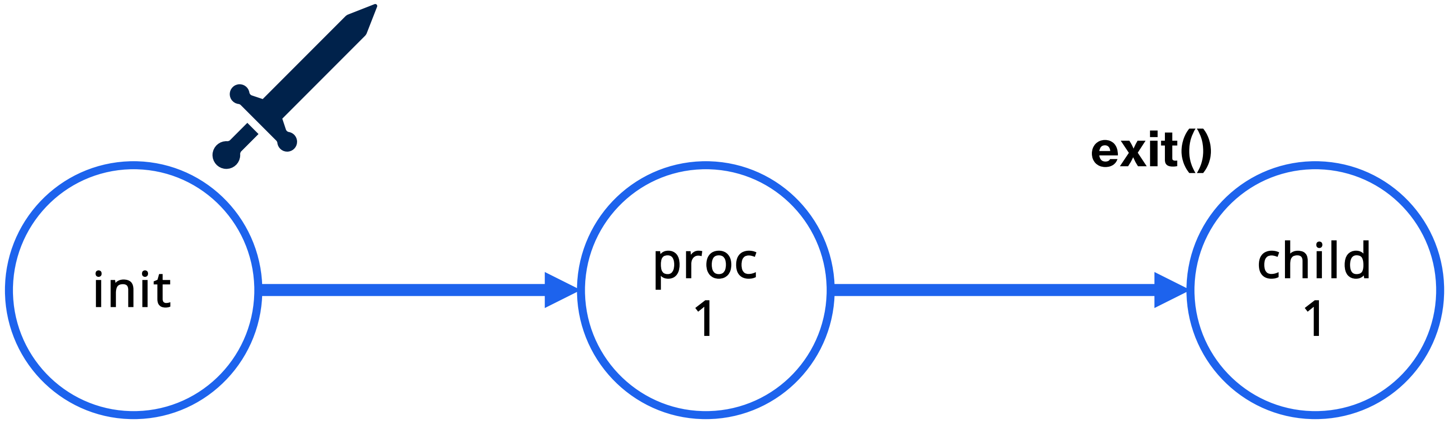
docker inspect **docker ps** **docker log id**

FIN

Expansión

Problema del PID1

Procesos Zombies



Soluciones para PID1

Soluciones (Reap problem)

- init (Unix)
- bash (no gestiona signals)
- supervisor
- phusion/baseimage
- dumb-init
- docker run --init / init: true
- tini

Dockerfile (VI)

Cachéame

[CACHED] FROM . . .

[CACHED] COPY . . .

[CACHED] RUN . . .

RUN . . .

CMD . . .

Dockerfile (VII)

Multistage

```
FROM alpine:latest AS builder
RUN apk --no-cache add build-base

FROM builder AS building_image
COPY src source.cpp
RUN g++ src/*.c
```

```
COPY --from=0
COPY --from=builder
```

Dockerfile (VIII)

pipefail

```
command_1 | command_2
command_1 | command_2
```

```
RUN set -o pipefail && command_1 | command_2
```

Dockerfile (IX)

scripts

```
#!/bin/bash

set -e

command_1
command_2
command_3
```

Usuarios

Anti root

```
#Cambiar usuario
USER usuario
```

Rootless

Docker Scout

Cuidando las vulnerabilidades

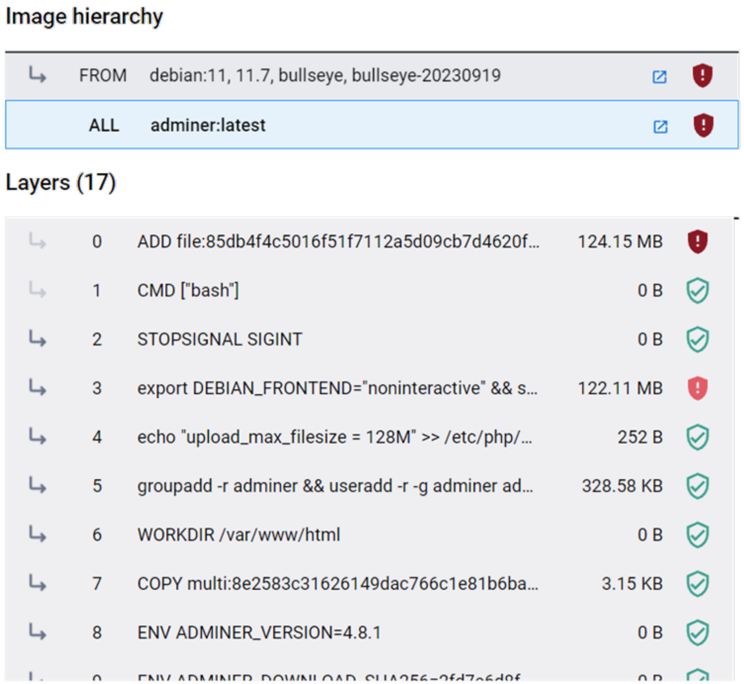
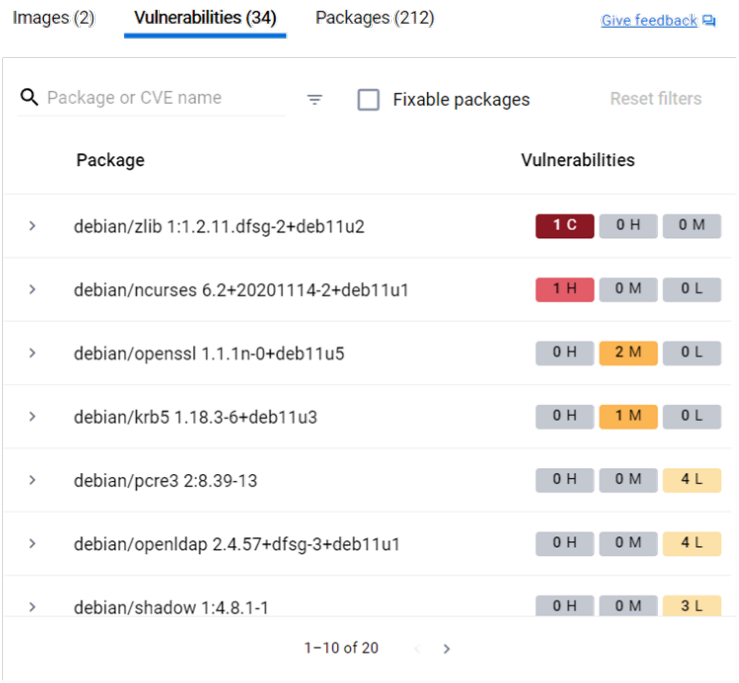


Image hierarchy

FROM	debian:11, 11.7, bullseye, bullseye-20230919	↗	!
ALL	adminer:latest	↗	!

Layers (17)

0	ADD file:85db4f4c5016f51f7112a5d09cb7d4620f...	124.15 MB	!
1	CMD ["bash"]	0 B	✓
2	STOPSIGNAL SIGINT	0 B	✓
3	export DEBIAN_FRONTEND="noninteractive" && s...	122.11 MB	!
4	echo "upload_max_filesize = 128M" >> /etc/php/...	252 B	✓
5	groupadd -r adminer && useradd -r -g adminer ad...	328.58 KB	✓
6	WORKDIR /var/www/html	0 B	✓
7	COPY multi:8e2583c31626149dac766c1e81b6ba...	3.15 KB	✓
8	ENV ADMINER_VERSION=4.8.1	0 B	✓
9	ENV ADMINER_DOWNLOAD_SHA256=3fd7e6d8f...	0 B	✓



Package	Vulnerabilities		
debian/zlib 1:1.2.11.dfsg-2+deb11u2	1 C	0 H	0 M
debian/ncurses 6.2+20201114-2+deb11u1	1 H	0 M	0 L
debian/openssl 1.1.1n-0+deb11u5	0 H	2 M	0 L
debian/krb5 1.18.3-6+deb11u3	0 H	1 M	0 L
debian/pcre3 2:8.39-13	0 H	0 M	4 L
debian/openldap 2.4.57+dfsg-3+deb11u1	0 H	0 M	4 L
debian/shadow 1:4.8.1-1	0 H	0 M	3 L

1–10 of 20

Secrets

secrets

Fuera

Dentro

```
services
 abc:
   secrets:
     - db_password


secrets:
   db_password:
     file: db_password.txt
```

Networks (II)

Configurando drivers

- ***bridge***, (default), red privada
- ***host***, *red del host*
- ***overlay***, *entre hosts (swarm)*
- ***macvlan***, *red física*
- ***none***, *aislado*

Volumes

Configurando volúmenes^3

- ***local***, *almacén en host (driver)*
- ***nfs***, *volumen desde sistemas NFS*
- ***bind***, *enlazar directorios*
- ***volume***, *en volúmenes Docker*
- ***tmpfs***, *en RAM (temporal)*
- ***azure_file / efs***, *en servicios de la nube*

Docker Compose Up

Cosas que pasan (a veces)

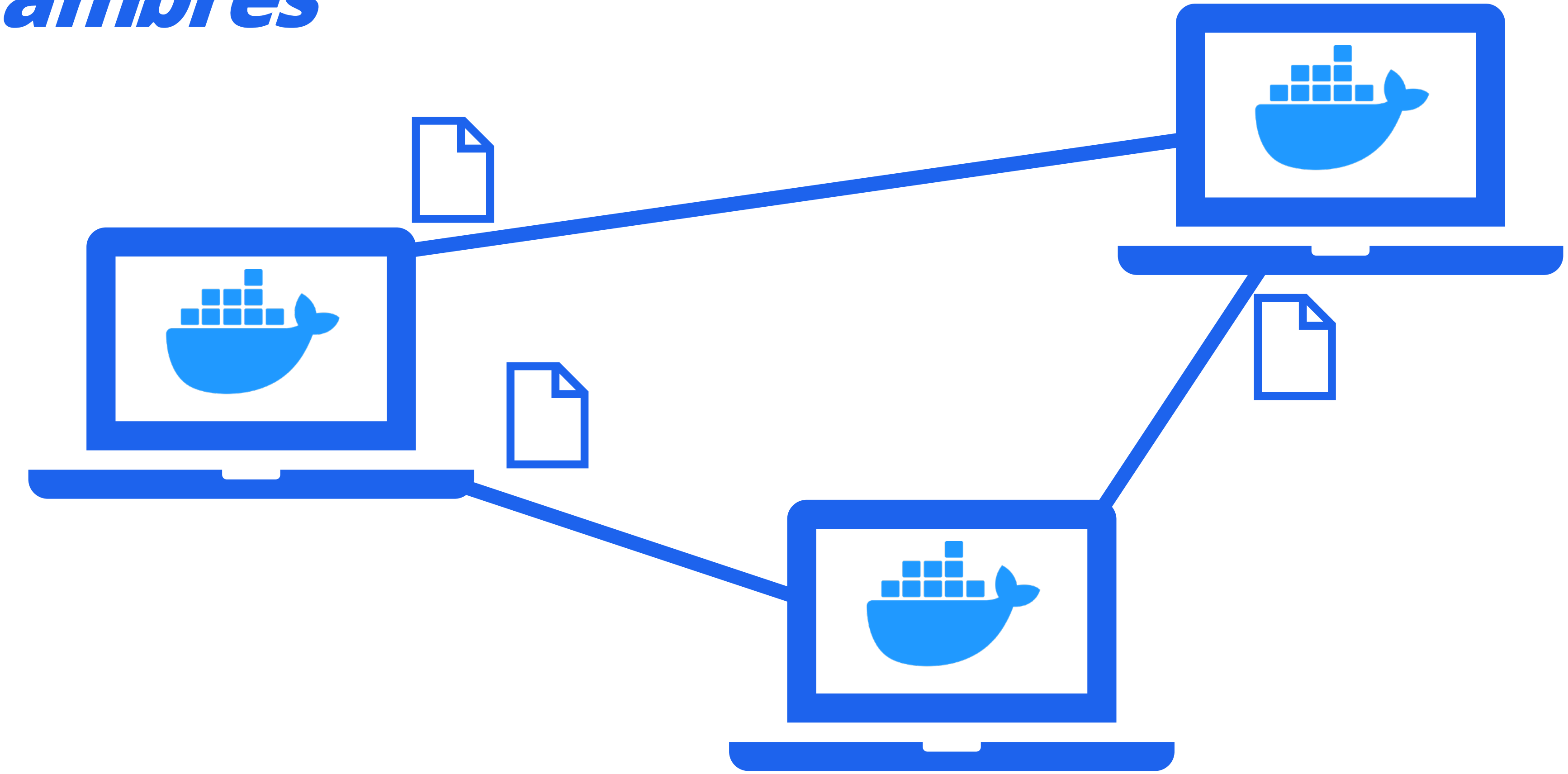
```
docker compose up # Con argumento build
# Imagen y no se actualiza

docker compose up --build
# Se creó la imagen y no se actualiza

docker compose up --build --force-recreate
# Se crea la imagen y reinicia el contenedor
```

Docker Machine

Docker Swarm

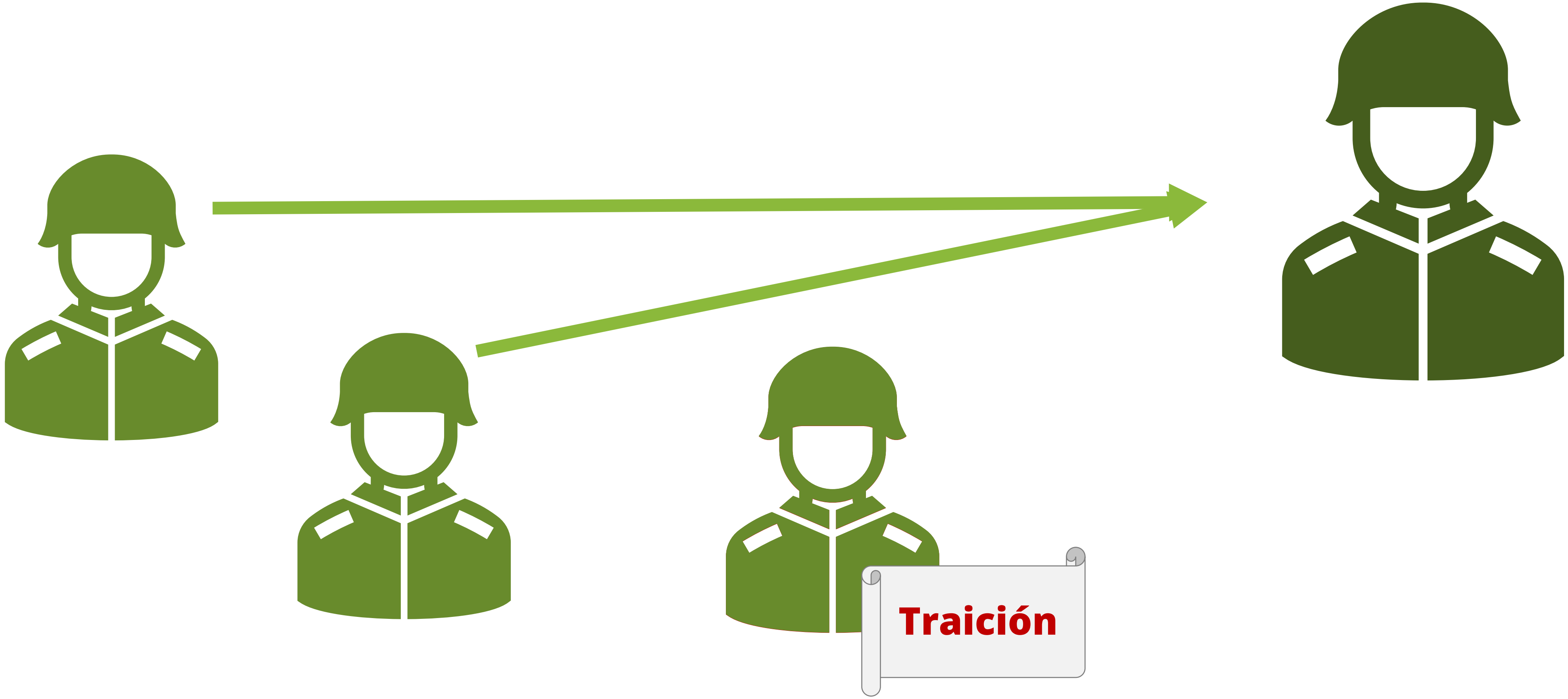
Docker Swarm (I)

Enjambres



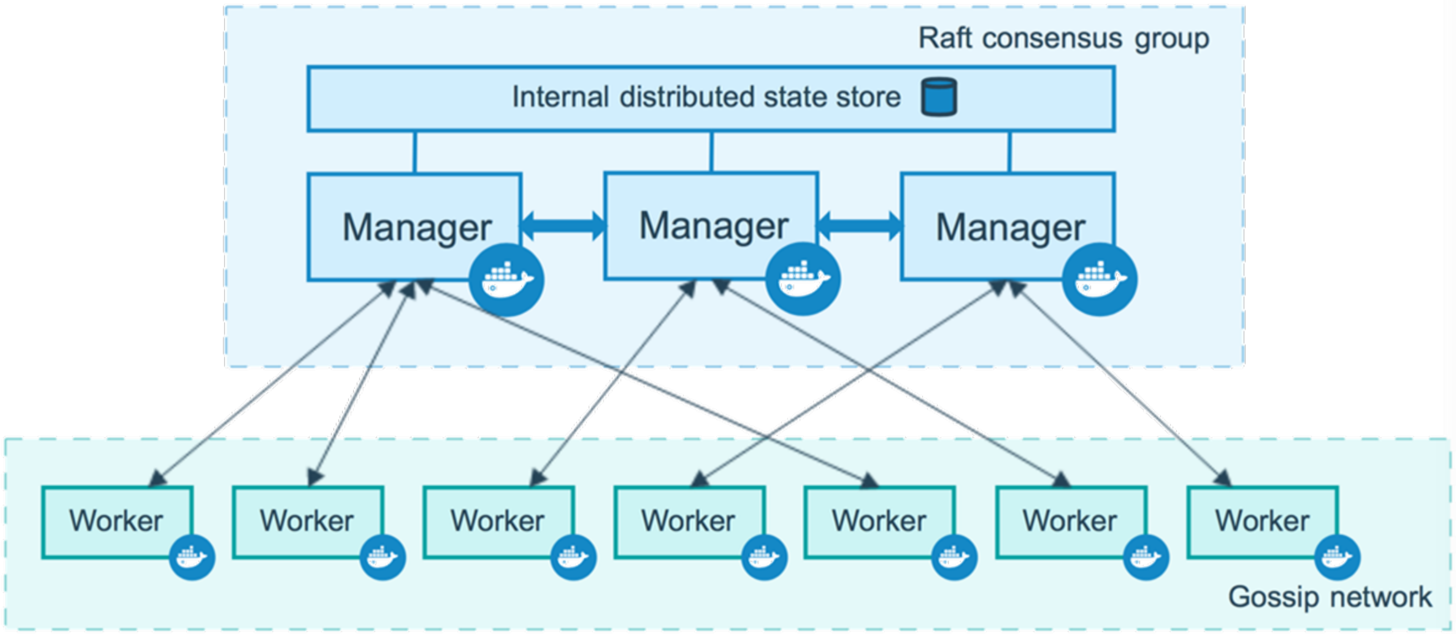
Docker Swarm (II)

Bizantinos



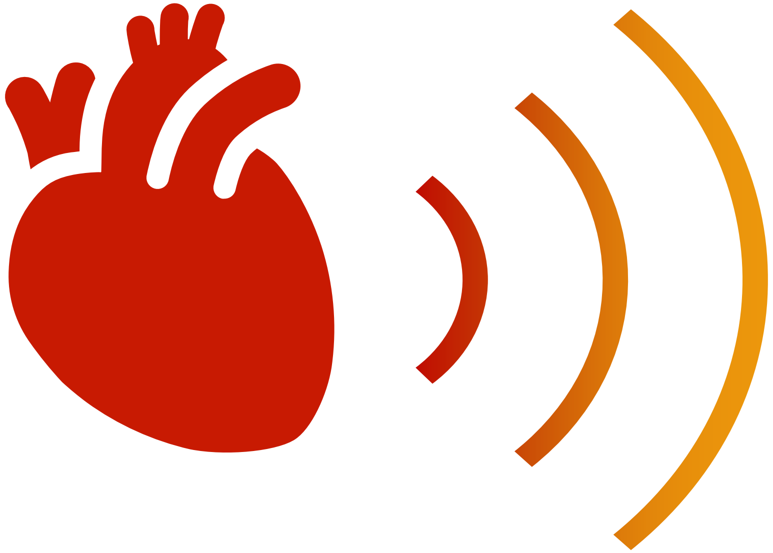
Docker Swarm (III)

Consenso



Docker Swarm (IV)

Heartbeat



Docker Swarm (V)

Docker Compose

```yaml
deploy:
  mode: replicated
  replicas: 2
  restart_policy:
    condition: on-failure
```

Docker Swarm (VI)

Documentación (otra vez)

❗ Note when using docker stack deploy . . .

Docker Swarm (VII)

Dándole a la colmena

```
docker swarm init
docker swarm join --token unTokenMuyLargo
docker stack deploy
```

```
docker service ls
```

```
docker node ls
```

THE END

Bibliografía y Recursos

https://docs.docker.com/
https://docs.docker.com/develop/develop-images/dockerfile_best-practices/
https://www.docker.com/resources/what-container/
https://learn.microsoft.com/es-es/windows/images/vscode-remote-containers.png
https://github.githubassets.com/images/modules/site/social-cards/codespaces-ga-individuals.jpg
https://seeklogo.com/images/S/scratch-cat-logo-7F652C6253-seeklogo.com.png
https://docs.docker.com/engine/swarm/images/swarm-diagram.png
https://www.docker.com/wp-content/uploads/2023/07/scout-logo-white-new.svg
https://www.cvedetails.com/vulnerability-list.php?vendor_id=13534&product_id=28125
https://hub.docker.com/extensions/docker/disk-usage-extension
De Oracle Corporation - This image may be found in VirtualBox 4.2 for Windows hosts, GPLv2,
https://commons.wikimedia.org/w/index.php?curid=24112652