GUÍA DOCKER

Práctica: Guía Docker.
Fecha: 20/02/24.
Autor/a: BUDDY.

IPs:	SO:	Equipos:	Finalidad:	Páginas:
enhypen: 172.16.1.30	Linux (lxc)	**enhypen** (instalar y gestionar Docker)	Conocer a fondo Docker	—---

ÍNDICE:

0. Introducción

Docker es una plataforma de código abierto que **permite automatizar el despliegue de aplicaciones dentro de contenedores** de software. Los **contenedores son unidades de software ligeros y portátiles que incluyen todo lo necesario para ejecutar una aplicación**: código, bibliotecas, herramientas y configuraciones. Docker proporciona **una forma de empaquetar y distribuir aplicaciones junto con todas sus dependencias en un contenedor único**, que se puede ejecutar en cualquier entorno compatible con Docker.

Esta es una **guía completa sobre Docker**, para saber más sobre las imágenes, los volúmenes, los comandos necesarios para instalar Docker en un lxc, etc.
Todo lo que necesitas saber de Docker, lo encontrarás aquí.

1. Actualizar el sistema

Comenzamos actualizando el sistema para evitar cualquier problema con los paquetes obsoletos. Ejecutamos el comando: ***"sudo apt update && sudo apt upgrade"***.

```
root@enhypen:~# sudo apt update && sudo apt upgrade
Hit:1 http://archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:3 http://archive.ubuntu.com/ubuntu jammy-security InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
All packages are up to date.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Calculating upgrade... Done
Get more security updates through Ubuntu Pro with 'esm-apps' enabled:
  gsasl-common libgsasl7
Learn more about Ubuntu Pro at https://ubuntu.com/pro
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
root@enhypen:~#
```

2. Instalar Docker

Desinstalamos cualquier intento previo de instalar Docker para que no cree conflicto con la instalación: ***"for pkg in docker.io docker-doc docker-compose docker-compose-v2 podman-docker containerd runc; do sudo apt-get remove $pkg; done"***.

```
root@enhypen:~# for pkg in docker.io docker-doc docker-compose docker-compose-v2 podman-docker containerd runc; do sudo apt-get remove $pkg; done
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Package 'docker.io' is not installed, so not removed
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Package 'docker-doc' is not installed, so not removed
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Package 'docker-compose' is not installed, so not removed
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Package 'docker-compose-v2' is not installed, so not removed
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Package 'podman-docker' is not installed, so not removed
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Package 'containerd' is not installed, so not removed
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
```

Después de eso, lo instalaremos utilizando los repositorios apt, para ello vamos a ejecutar este comando: ***"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".

Con eso hemos añadido la clave GPG oficial de Docker.

```
root@enhypen:~# # Add Docker's official GPG key:
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

# Add the repository to Apt sources:
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu \
  $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update
Get:1 https://download.docker.com/linux/ubuntu jammy InRelease [48.8 kB]
Hit:2 http://archive.ubuntu.com/ubuntu jammy InRelease
Get:3 http://archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Get:4 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages [26.7 kB]
Get:5 http://archive.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Get:6 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [1377 kB]
Fetched 1682 kB in 2s (799 kB/s)
Reading package lists... Done
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ca-certificates is already the newest version (20230311ubuntu0.22.04.1).
curl is already the newest version (7.81.0-1ubuntu1.15).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
Hit:1 https://download.docker.com/linux/ubuntu jammy InRelease
Hit:2 http://archive.ubuntu.com/ubuntu jammy InRelease
Hit:3 http://archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:4 http://archive.ubuntu.com/ubuntu jammy-security InRelease
```

Y con este otro comando, añadimos apt a los repositorios fuente: ***“echo ***
"deb [arch=$(dpkg --print-architecture)
signed-by=/etc/apt/keyrings/docker.asc]
***https://download.docker.com/linux/ubuntu ***
***$(. /etc/os-release && echo "$VERSION_CODENAME") stable" | ***
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update”.

```
# Add the repository to Apt sources:
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu \
  $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update
Get:1 https://download.docker.com/linux/ubuntu jammy InRelease [48.8 kB]
Hit:2 http://archive.ubuntu.com/ubuntu jammy InRelease
Get:3 http://archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Get:4 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages [26.7 kB]
Get:5 http://archive.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Get:6 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [1377 kB]
Fetched 1682 kB in 2s (799 kB/s)
Reading package lists... Done
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ca-certificates is already the newest version (20230311ubuntu0.22.04.1).
curl is already the newest version (7.81.0-1ubuntu1.15).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
Hit:1 https://download.docker.com/linux/ubuntu jammy InRelease
Hit:2 http://archive.ubuntu.com/ubuntu jammy InRelease
Hit:3 http://archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:4 http://archive.ubuntu.com/ubuntu jammy-security InRelease
Reading package lists... Done
```

Instalamos los paquetes Docker: ***“sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin”***.

```
root@enhypen:~# sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
containerd.io is already the newest version (1.6.28-1).
docker-buildx-plugin is already the newest version (0.12.1-1~ubuntu.22.04~jammy).
docker-ce-cli is already the newest version (5:25.0.3-1~ubuntu.22.04~jammy).
docker-ce is already the newest version (5:25.0.3-1~ubuntu.22.04~jammy).
docker-compose-plugin is already the newest version (2.24.5-1~ubuntu.22.04~jammy).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
```

Ahora verificamos que Docker Engine se ha instalado con el comando: ***“sudo docker run hello-world”***.

```
root@enhypen:~# sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
c1ec31eb5944: Pull complete
Digest: sha256:d000bc569937abbe195e20322a0bde6b2922d805332fd6d8a68b19f524b7d21d
Status: Downloaded newer image for hello-world:latest

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

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

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

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

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

Aquí vemos que Docker se ha instalado correctamente.

3. Crear una imagen

Primero hemos de clonar este repositorio (que es básicamente una app de Docker donde poder crear nuestros contenedores, imágenes, volúmenes, etc).

Ejecutamos el comando: ***"git clone https://github.com/docker/welcome-to-docker"***.

```
root@enhypen:~# git clone https://github.com/docker/welcome-to-docker
Cloning into 'welcome-to-docker'...
remote: Enumerating objects: 125, done.
remote: Counting objects: 100% (58/58), done.
remote: Compressing objects: 100% (36/36), done.
remote: Total 125 (delta 33), reused 34 (delta 19), pack-reused 67
Receiving objects: 100% (125/125), 319.54 KiB | 1.10 MiB/s, done.
Resolving deltas: 100% (55/55), done.
root@enhypen:~#
```

Una vez clonado el repositorio, vamos a navegar por el directorio:

```
root@enhypen:~# ls
welcome-to-docker
root@enhypen:~# cd welcome-to-docker
root@enhypen:~/welcome-to-docker# ls
Dockerfile  MAINTAINERS.md  README.md  package-lock.json  package.json  public  src
root@enhypen:~/welcome-to-docker#
```

Dentro del directorio vemos que hay un fichero Dockerfile, ese fichero Dockerfile contiene lo necesario para crear una imagen.

```
  GNU nano 6.2                                    Dockerfile
# Start your image with a node base image
FROM node:18-alpine

# The /app directory should act as the main application directory
WORKDIR /app

# Copy the app package and package-lock.json file
COPY package*.json ./

# Copy local directories to the current local directory of our docker image (/app)
COPY ./src ./src
COPY ./public ./public

# Install node packages, install serve, build the app, and remove dependencies at the end
RUN npm install \
    && npm install -g serve \
    && npm run build \
    && rm -fr node_modules

EXPOSE 3000

# Start the app using serve command
CMD [ "serve", "-s", "build" ]
```

Podemos ver que es una imagen de Node, la versión 18.

Que el directorio en el que se va a trabajar será ***"/app"***.

Podemos ver que le dice que copie el paquete de la app y el paquete .json.

Que copie los directorios locales al directorio local actual de nuestra imagen Docker.

Especifica también que instale los paquetes de Node para construir la app y eliminar las dependencias al final.

Por último, especifica que arranque la app utilizando el comando "serve".

Para crear una imagen, nos situamos en el directorio de la app y ejecutamos el comando: ***"docker build -t welcome-to-docker ."***.

```
root@enhypen:~/welcome-to-docker# ls
Dockerfile  MAINTAINERS.md  README.md  package-lock.json  package.json  public  src
root@enhypen:~/welcome-to-docker# docker build -t welcome-to-docker .
[+] Building 4.9s (4/10)                                                    docker:default
 => [internal] load build definition from Dockerfile                                 0.1s
 => => transferring dockerfile: 647B                                                 0.0s
 => [internal] load metadata for docker.io/library/node:18-alpine                    1.5s
 => [internal] load .dockerignore                                                    0.0s
 => => transferring context: 52B                                                     0.0s
 => [1/6] FROM docker.io/library/node:18-alpine@sha256:ca9f6cb0466f9638e59e0c249d335a07c867cd50c429b5c7  3.3s
 => => resolve docker.io/library/node:18-alpine@sha256:ca9f6cb0466f9638e59e0c249d335a07c867cd50c429b5c7  0.0s
 => => sha256:4abcf20661432fb2d719aaf90656f55c287f8ca915dc1c92ec14ff61e67fbaf8 1.05MB / 3.41MB  3.2s
 => => sha256:e7ced292c644a1f7bc776dcc401164b67c9224f8592cc83b8c42e237668a0c7f 2.10MB / 40.25MB  3.2s
 => => sha256:b32c0114bba5af3e85af37dbc23b1e026850aba590099b81bf75946327b3a9e8 0B / 2.34MB  3.2s
 => => sha256:ca9f6cb0466f9638e59e0c249d335a07c867cd50c429b5c7830dda1bed584649 1.43kB / 1.43kB  0.0s
 => => sha256:affdf979bd8ec516bf189d451b8ac68dd50adc49adc4c4014963556c11efeda4 1.16kB / 1.16kB  0.0s
 => => sha256:24d8fcd7167fb06e91dc7228311105013dc042f6875ff2528ff7a41c04770112 7.14kB / 7.14kB  0.0s
 => [internal] load build context                                                    0.1s
 => => transferring context: 691.39kB                                                0.1s
```

Después de un buen rato, terminará.

```
[+] Building 526.3s (11/11) FINISHED                                        docker:default
 => [internal] load build definition from Dockerfile                                 0.1s
 => => transferring dockerfile: 647B                                                 0.0s
 => [internal] load metadata for docker.io/library/node:18-alpine                    1.5s
 => [internal] load .dockerignore                                                    0.0s
 => => transferring context: 52B                                                     0.0s
 => [1/6] FROM docker.io/library/node:18-alpine@sha256:ca9f6cb0466f9638e59e0c249d335a07c867cd50c429b5c  29.8s
 => => resolve docker.io/library/node:18-alpine@sha256:ca9f6cb0466f9638e59e0c249d335a07c867cd50c429b5c7  0.0s
 => => sha256:4abcf20661432fb2d719aaf90656f55c287f8ca915dc1c92ec14ff61e67fbaf8 3.41MB / 3.41MB  11.9s
 => => sha256:e7ced292c644a1f7bc776dcc401164b67c9224f8592cc83b8c42e237668a0c7f 40.25MB / 40.25MB  25.5s
 => => sha256:b32c0114bba5af3e85af37dbc23b1e026850aba590099b81bf75946327b3a9e8 2.34MB / 2.34MB  13.9s
 => => sha256:ca9f6cb0466f9638e59e0c249d335a07c867cd50c429b5c7830dda1bed584649 1.43kB / 1.43kB  0.0s
 => => sha256:affdf979bd8ec516bf189d451b8ac68dd50adc49adc4c4014963556c11efeda4 1.16kB / 1.16kB  0.0s
 => => sha256:24d8fcd7167fb06e91dc7228311105013dc042f6875ff2528ff7a41c04770112 7.14kB / 7.14kB  0.0s
 => => extracting sha256:4abcf20661432fb2d719aaf90656f55c287f8ca915dc1c92ec14ff61e67fbaf8  0.3s
 => => sha256:f3748d9674b0ca905fe23e1cb4ad0e49d6a605dbbfb9d0cf485f300a03f1eeff 450B / 450B  12.3s
 => => extracting sha256:e7ced292c644a1f7bc776dcc401164b67c9224f8592cc83b8c42e237668a0c7f  3.3s
 => => extracting sha256:b32c0114bba5af3e85af37dbc23b1e026850aba590099b81bf75946327b3a9e8  0.2s
 => => extracting sha256:f3748d9674b0ca905fe23e1cb4ad0e49d6a605dbbfb9d0cf485f300a03f1eeff  0.0s
 => [internal] load build context                                                    0.1s
 => => transferring context: 691.39kB                                                0.1s
 => [2/6] WORKDIR /app                                                               0.1s
 => [3/6] COPY package*.json ./                                                      0.1s
 => [4/6] COPY ./src ./src                                                           0.1s
 => [5/6] COPY ./public ./public                                                     0.1s
 => [6/6] RUN npm install     && npm install -g serve     && npm run build     && rm -fr node_modules  483.6s
 => exporting to image                                                              10.3s
 => => exporting layers                                                             10.3s
 => => writing image sha256:9579ba5d5d5104e2741453c80af5fbc8f811cde3dd0d78f70393ed2774a21176  0.0s
 => => naming to docker.io/library/welcome-to-docker                                 0.0s
root@enhypen:~/welcome-to-docker#
```

Si hacemos ***"docker images"***, podremos ver las imágenes creadas:

```
root@enhypen:~/welcome-to-docker# docker images
REPOSITORY          TAG       IMAGE ID       CREATED              SIZE
welcome-to-docker   latest    9579ba5d5d51   About a minute ago   226MB
hello-world         latest    d2c94e258dcb   9 months ago         13.3kB
root@enhypen:~/welcome-to-docker#
```

4. Arrancar una imagen

Para arrancar una imagen basta con ejecutar el comando: ***"docker run -d (id_container)"***, en este caso: ***"docker run -d 957"***.

```
root@enhypen:~/welcome-to-docker# docker run -d 957 welcome-to-docker
14be20dd6a6e949731d40a5771d09ef1af802413e2d051c3c1d5c35350a6e130
root@enhypen:~/welcome-to-docker#
```

Con el comando: ***"docker ps"*** podemos ver los contenedores que están en marcha.

```
root@enhypen:~/welcome-to-docker# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS         PORTS      NAMES
a20acd6c8e07   957       "docker-entrypoint.s…"   4 seconds ago   Up 2 seconds   3000/tcp   eager_chatterjee
root@enhypen:~/welcome-to-docker#
```

Podemos también acceder al contenedor con el comando: ***"docker exec -it (nombre_container) /bin/bash"***.

```
root@enhypen:~/welcome-to-docker# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS         PORTS      NAMES
a20acd6c8e07   957       "docker-entrypoint.s…"   4 minutes ago   Up 4 minutes   3000/tcp   eager_chatterjee
root@enhypen:~/welcome-to-docker# docker exec -it eager_chatterjee /bin/bash
OCI runtime exec failed: exec failed: unable to start container process: exec: "/bin/bash": stat /bin/bash: no
 such file or directory: unknown
root@enhypen:~/welcome-to-docker# docker exec -it eager_chatterjee sh
/app #
```

Habrá veces que no tengan /bin/bash, pero se puede probar con "sh" o "ls".

Nos podemos mover por los directorios:

```
root@enhypen:~/welcome-to-docker# docker exec -it eager_chatterjee sh
/app # ls
build              package-lock.json  package.json       public             src
/app # cd src
/app/src # ls
App.css      App.js       Confetti.js  index.css    index.js
/app/src #
```

5. Lanzar una app

Ejecutamos la app con el comando: ***"docker run -d -p 8088:80 welcome-to-docker docker/welcome-to-docker"***.

```
root@enhypen:~/welcome-to-docker# docker run -d -p 8088:80 welcome-to-docker docker/welcome-t
o-docker
1276e9b39bf32e13ea692c1ed95ab6d84bab0d0c6d2c995b133e07a90e148c71
root@enhypen:~/welcome-to-docker#
```

Estamos en un contenedor, por lo que no podemos ver la app dado que no hay navegador, así que tenemos que crear una regla en pfSense para poder entrar desde otro equipo que si tenga navegador.

6. Regla pfSense

Edit Redirect Entry

Disabled	☐ Disable this rule
No RDR (NOT)	☐ Disable redirection for traffic matching this rule This option is rarely needed. Don't use this without thorough knowledge of the implications.
Interface	WAN Choose which interface this rule applies to. In most cases "WAN" is specified.
Address Family	IPv4 Select the Internet Protocol version this rule applies to.
Protocol	TCP/UDP Choose which protocol this rule should match. In most cases "TCP" is specified.
Source	Display Advanced
Destination	☐ Invert match. WAN address (Type) / (Address/mask)
Destination port range	Other (From port) 52033 (Custom) Other (To port) 52033 (Custom) Specify the port or port range for the destination of the packet for this mapping. The 'to' field may be left empty if only mapping a single port.
Redirect target IP	Address or Alias (Type) 172.16.1.30 (Address)

Redirect target IP	Address or Alias (Type) 172.16.1.30 (Address) Enter the internal IP address of the server on which to map the ports. e.g.: 192.168.1.12 for IPv4 In case of IPv6 addresses, in must be from the same "scope", i.e. it is not possible to redirect from link-local addresses scope (fe80:*) to local scope (::1)
Redirect target port	SSH (Port) (Custom) Specify the port on the machine with the IP address entered above. In case of a port range, specify the beginning port of the range (the end port will be calculated automatically). This is usually identical to the "From port" above.
Description	Acceso SSH a app docker default A description may be entered here for administrative reference (not parsed).
No XMLRPC Sync	☐ Do not automatically sync to other CARP members This prevents the rule on Master from automatically syncing to other CARP members. This does NOT prevent the rule from being overwritten on Slave.
NAT reflection	Use system default
Filter rule association	Rule NAT Acceso SSH a app docker View the filter rule

7. Ejecutar múltiples contenedores

Para ejecutar múltiples contenedores, en este caso vamos a tomar los repositorios de Docker oficiales y los vamos a clonar. Para poder clonar dichos repositorios, hemos de ejecutar el comando: ***"git clone https://github.com/docker/multi-container-app"***.

```
root@enhypen:/# git clone https://github.com/docker/multi-container-app
Cloning into 'multi-container-app'...
remote: Enumerating objects: 2291, done.
remote: Counting objects: 100% (23/23), done.
remote: Compressing objects: 100% (17/17), done.
remote: Total 2291 (delta 5), reused 20 (delta 5), pack-reused 2268
Receiving objects: 100% (2291/2291), 3.83 MiB | 1.25 MiB/s, done.
Resolving deltas: 100% (360/360), done.
root@enhypen:/#
```

Navegamos un poco y encontramos el fichero importante, ***"compose.yaml"***.

```
root@enhypen:/# ls
bin   home   libx32       multi-container-app  run   tmp
boot  lib    lost+found   opt                  sbin  usr
dev   lib32  media        proc                 srv   var
etc   lib64  mnt          root                 sys
root@enhypen:/# cd multi-container-app
root@enhypen:/multi-container-app# ls
README.md  app  compose.yaml
root@enhypen:/multi-container-app#
```

Si abrimos el fichero compose.yaml, encontraremos lo siguiente.

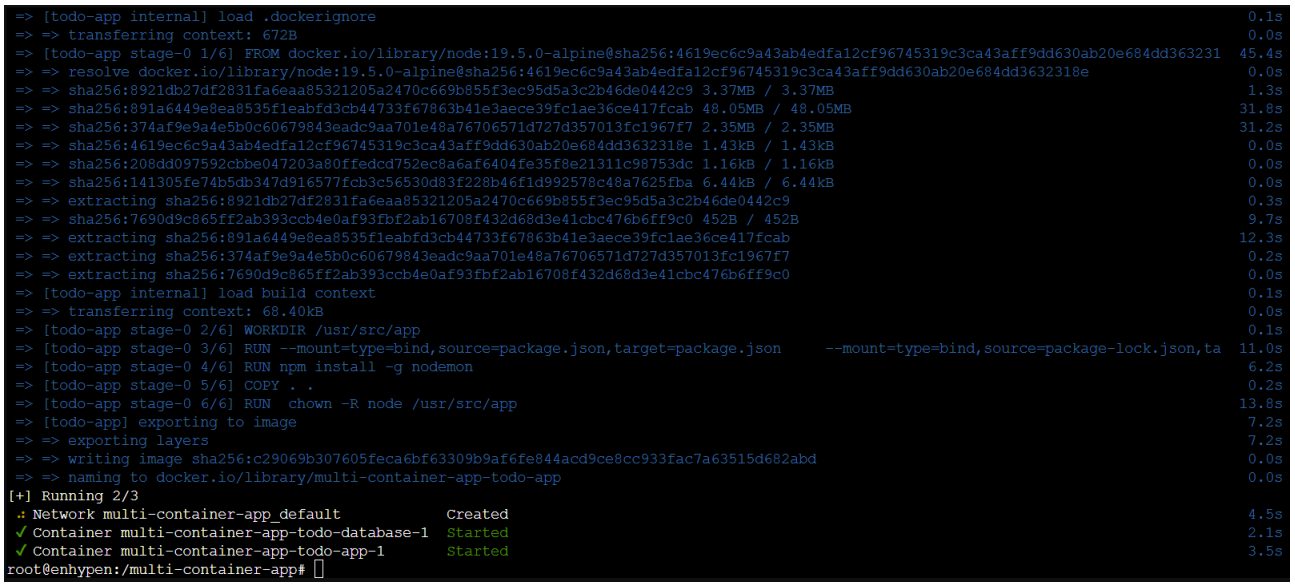
```
  GNU nano 6.2                                                compose.yaml

services:
  todo-app:
    build:
      context: ./app
    depends_on:
      - todo-database
    environment:
      NODE_ENV: production
    ports:
      - 3000:3000
      - 35729:35729
    develop:
      watch:
        - path: ./app/package.json
          action: rebuild
        - path: ./app
          target: /usr/src/app
          action: sync

  todo-database:
    image: mongo:6
    #volumes:
    #  - database:/data/db
    ports:
      - 27017:27017
```

Ahora, dentro de la app, ejecutamos ***"docker compose up -d"***.

```
root@enhypen:/# ls
bin   home   libx32      multi-container-app  run   tmp
boot  lib    lost+found  opt                  sbin  usr
dev   lib32  media       proc                 srv   var
etc   lib64  mnt         root                 sys
root@enhypen:/# cd multi-container-app
root@enhypen:/multi-container-app# docker compose up -d
[+] Running 4/9
 ⠧ todo-database 8 layers [ ⣿⣿⣿⣿⣿   ] 3.709MB/29.54MB Pulling                                                    4.8s
   ⠧ 01007420e9b0 Downloading [======>                                          ]  3.709MB/29.54MB                 2.6s
   ✓ c9ca97e2c467 Download complete                                                                                0.5s
   ✓ bd242b4fa75f Download complete                                                                                1.7s
   ✓ d9608f6bb829 Download complete                                                                                2.3s
   ✓ dfb180c9e7b5 Download complete                                                                                2.5s
   ⠧ 44fe9b1d0ed3 Waiting                                                                                          2.6s
   ⠧ 6d50a4b3eea6 Waiting                                                                                          2.6s
   ⠧ e5f2ecffbdb8 Waiting                                                                                          2.6s
```


=> [todo-app internal] load .dockerignore 0.1s
=> => transferring context: 672B 0.0s
=> [todo-app stage-0 1/6] FROM docker.io/library/node:19.5.0-alpine@sha256:4619ec6c9a43ab4edfa12cf96745319c3ca43aff9dd630ab20e684dd363231 45.4s
=> => resolve docker.io/library/node:19.5.0-alpine@sha256:4619ec6c9a43ab4edfa12cf96745319c3ca43aff9dd630ab20e684dd3632318e 0.0s
=> => sha256:8921db27df2831fa6eaa85321205a2470c669b855f3ec95d5a3c2b46de0442c9 3.37MB / 3.37MB 1.3s
=> => sha256:891a6449e8ea8535f1eabfd3cb44733f67863b41e3aece39fc1ae36ce417fcab 48.05MB / 48.05MB 31.8s
=> => sha256:374af9e9a4e5b0c60679843eadc9aa701e48a76706571d727d357013fc1967f7 2.35MB / 2.35MB 31.2s
=> => sha256:4619ec6c9a43ab4edfa12cf96745319c3ca43aff9dd630ab20e684dd3632318e 1.43kB / 1.43kB 0.0s
=> => sha256:208dd097592cbbe047203a80ffedcd752ec8a6af6404fe35f8e21311c98753dc 1.16kB / 1.16kB 0.0s
=> => sha256:141305fe74b5db347d916577fcb3c56530d83f228b46f1d992578c48a7625fba 6.44kB / 6.44kB 0.0s
=> => extracting sha256:8921db27df2831fa6eaa85321205a2470c669b855f3ec95d5a3c2b46de0442c9 0.3s
=> => sha256:7690d9c865ff2ab393ccb4e0af93fbf2ab16708f432d68d3e41cbc476b6ff9c0 452B / 452B 9.7s
=> => extracting sha256:891a6449e8ea8535f1eabfd3cb44733f67863b41e3aece39fc1ae36ce417fcab 12.3s
=> => extracting sha256:374af9e9a4e5b0c60679843eadc9aa701e48a76706571d727d357013fc1967f7 0.2s
=> => extracting sha256:7690d9c865ff2ab393ccb4e0af93fbf2ab16708f432d68d3e41cbc476b6ff9c0 0.0s
=> [todo-app internal] load build context 0.1s
=> => transferring context: 68.40kB 0.0s
=> [todo-app stage-0 2/6] WORKDIR /usr/src/app 0.1s
=> [todo-app stage-0 3/6] RUN --mount=type=bind,source=package.json,target=package.json --mount=type=bind,source=package-lock.json,ta 11.0s
=> [todo-app stage-0 4/6] RUN npm install -g nodemon 6.2s
=> [todo-app stage-0 5/6] COPY . . 0.2s
=> [todo-app stage-0 6/6] RUN chown -R node /usr/src/app 13.8s
=> [todo-app] exporting to image 7.2s
=> => exporting layers 7.2s
=> => writing image sha256:c29069b307605feca6bf63309b9af6fe844acd9ce8cc933fac7a63515d682abd 0.0s
=> => naming to docker.io/library/multi-container-app-todo-app 0.0s
[+] Running 2/3
⠿ Network multi-container-app_default Created 4.5s
✓ Container multi-container-app-todo-database-1 Started 2.1s
✓ Container multi-container-app-todo-app-1 Started 3.5s
root@enhypen:/multi-container-app#