

LABORATORIO DE SISTEMAS TELEMÁTICOS II



TRABAJO PRESENTADO A:

FULVIO YESID VIVAS CANTERO

PRESENTADO POR:

OSCAR EDUARDO ARIAS CARVAJAL

NESTOR JAIME ALEGRIA ALEGRIA

UNIVERSIDAD DEL CAUCA

FACULTAD DE INGENIERIA DE ELECTRONICA Y TELECOMUNICACIONES

POPAYÁN-CAUCA

2023

## INTRODUCCIÓN

En el mundo de la tecnología y el desarrollo de software, la contenerización se ha convertido en una herramienta esencial para simplificar la implementación y gestión de aplicaciones. Docker, una de las plataformas líderes en este campo, ha revolucionado la forma en que se construyen y ejecutan aplicaciones. Este informe explora varios conceptos clave en el contexto de Docker, incluyendo imágenes y contenedores, y cómo se utilizan para alojar aplicaciones, como el servidor web Apache. También se analiza la utilidad de la imagen "hello-world" y la importancia de los volúmenes en Docker, que permiten la persistencia de datos en un entorno de contenedorización. A lo largo de este informe, se destacarán las ventajas y casos de uso de estas tecnologías, proporcionando una visión general completa de su relevancia en el mundo del desarrollo y la administración de aplicaciones en contenedores.

## MARCO TEÓRICO

### DOCKER

Docker es una plataforma de contenerización que se utiliza para desarrollar, enviar y ejecutar aplicaciones en contenedores. Los contenedores son entornos ligeros y portátiles que contienen todas las dependencias necesarias para ejecutar una aplicación, incluyendo el código, bibliotecas, configuraciones y más. Docker simplifica la gestión de aplicaciones al encapsularlas en contenedores, lo que facilita la implementación y la escalabilidad de las aplicaciones en diferentes entornos, como entornos locales de desarrollo, servidores de producción o la nube.

A continuación, se presentan algunas características clave de Docker:

1. **Portabilidad:** Los contenedores Docker son altamente portátiles, lo que significa que puedes crear un contenedor en una máquina de desarrollo y ejecutarlo sin problemas en una máquina de producción o en la nube, siempre que Docker esté instalado.
2. **Aislamiento:** Docker utiliza tecnologías de aislamiento del sistema operativo, como namespaces y cgroups, para garantizar que los contenedores estén separados unos de otros y del sistema host, lo que evita conflictos de dependencias y posibles problemas de seguridad.
3. **Facilidad de uso:** Docker proporciona una interfaz de línea de comandos y una interfaz gráfica de usuario que facilita la creación, administración y gestión de contenedores.
4. **Versionamiento:** Docker permite la creación de imágenes de contenedor que se pueden versionar, lo que facilita la distribución y el control de versiones de aplicaciones y entornos de desarrollo.
5. **Escalabilidad:** Puedes escalar aplicaciones y servicios fácilmente mediante la replicación de contenedores en un clúster de contenedores gestionado.

6. **Integración:** Docker se integra bien con herramientas de automatización y orquestación, como Docker Compose, Kubernetes y otros, lo que facilita la administración de aplicaciones complejas y distribuidas.

En resumen, Docker es una tecnología ampliamente utilizada en el desarrollo de software y la administración de infraestructuras, ya que simplifica la implementación y gestión de aplicaciones, mejora la eficiencia y la portabilidad, y facilita la colaboración entre equipos de desarrollo y operaciones.

## IMAGEN HELLO-WORLD

La imagen "hello-world" en el contexto de Docker es una imagen extremadamente simple y básica que se utiliza comúnmente para verificar si la instalación de Docker está funcionando correctamente y puede ejecutar contenedores. Esta imagen se usa principalmente con fines de prueba y para familiarizarse con Docker.

Cuando ejecutas un contenedor basado en la imagen "hello-world", lo que hace es imprimir un mensaje en la terminal que dice "Hello from Docker!" junto con alguna información adicional, como la versión de Docker que estás utilizando.

El propósito principal de esta imagen es asegurarse de que Docker esté instalado y configurado correctamente en tu sistema. Si ves el mensaje "Hello from Docker!" y no hay errores, significa que Docker está funcionando correctamente. Es una especie de "hola mundo" para Docker que te permite realizar una prueba inicial de tu entorno Docker.

Es importante destacar que la imagen "hello-world" es muy pequeña y no contiene ninguna aplicación o servicio significativo; su único propósito es verificar el funcionamiento básico de Docker. Para trabajar con aplicaciones reales, necesitarás utilizar imágenes más específicas que contengan el software y las dependencias necesarias para tu aplicación.

## IMAGEN APACHE

Una imagen de Apache generalmente se refiere a una imagen de contenedor que contiene el servidor web Apache HTTP Server, comúnmente conocido simplemente como Apache. Apache HTTP Server es uno de los servidores web más populares y ampliamente utilizados en el mundo. Se utiliza para servir sitios web estáticos y dinámicos, aplicaciones web y otros recursos en la World Wide Web.

Una imagen de Apache típicamente contiene una instalación de Apache HTTP Server configurada y lista para ser ejecutada en un contenedor Docker u otro entorno de contenedorización. Estas imágenes suelen estar disponibles en el Docker Hub y otras plataformas de registro de imágenes de contenedor, lo que facilita su descarga y uso.

Al utilizar una imagen de Apache, puedes crear y ejecutar contenedores que sirvan aplicaciones web o sitios estáticos utilizando Apache como servidor web. Puedes personalizar la configuración de Apache, agregar tu código de sitio web y configurar el servidor según tus necesidades específicas.

La imagen "httpd" es una de las imágenes oficiales de Apache que puedes encontrar en Docker Hub, y "nombre\_de\_la\_version" debe ser reemplazado por la versión específica que deseas utilizar.

En resumen, una imagen de Apache en el contexto de la contenerización es una forma conveniente de distribuir y ejecutar el servidor web Apache en contenedores, lo que facilita la implementación y la administración de aplicaciones web y sitios en entornos de contenedor.

## VOLÚMENES

En Docker, los volúmenes son un mecanismo que permite que los datos persistan más allá del ciclo de vida de un contenedor. Los volúmenes en Docker son directorios o sistemas de archivos que son gestionados por Docker y que se utilizan para almacenar y compartir datos entre contenedores y el sistema host. Los volúmenes son una forma de garantizar la persistencia de datos, lo que significa que los datos dentro de un volumen no se perderán cuando un contenedor se detenga o se elimine.

Aquí hay algunas características clave de los volúmenes en Docker:

1. **Persistencia de Datos:** Los volúmenes permiten que los datos se mantengan de manera persistente, incluso después de que el contenedor que los utiliza se haya detenido o eliminado. Esto es útil para almacenar datos que deben conservarse entre ejecuciones de contenedores.
2. **Compartir Datos entre Contenedores:** Los volúmenes pueden ser compartidos entre múltiples contenedores, lo que facilita el intercambio de datos o configuraciones entre aplicaciones en contenedores diferentes.
3. **Separación de Datos y Contenido:** Utilizar volúmenes permite separar los datos de la aplicación y el contenido de los contenedores, lo que hace que los contenedores sean más efímeros y fáciles de administrar.
4. **Personalización de Rutas de Montaje:** Puedes especificar rutas de montaje personalizadas al utilizar volúmenes, lo que te brinda un mayor control sobre la ubicación de los datos en el sistema host.
5. **Compatibilidad con Copias de Seguridad:** Los volúmenes son una forma conveniente de respaldar y restaurar datos de contenedores, ya que puedes realizar copias de seguridad de los datos almacenados en un volumen de manera sencilla.

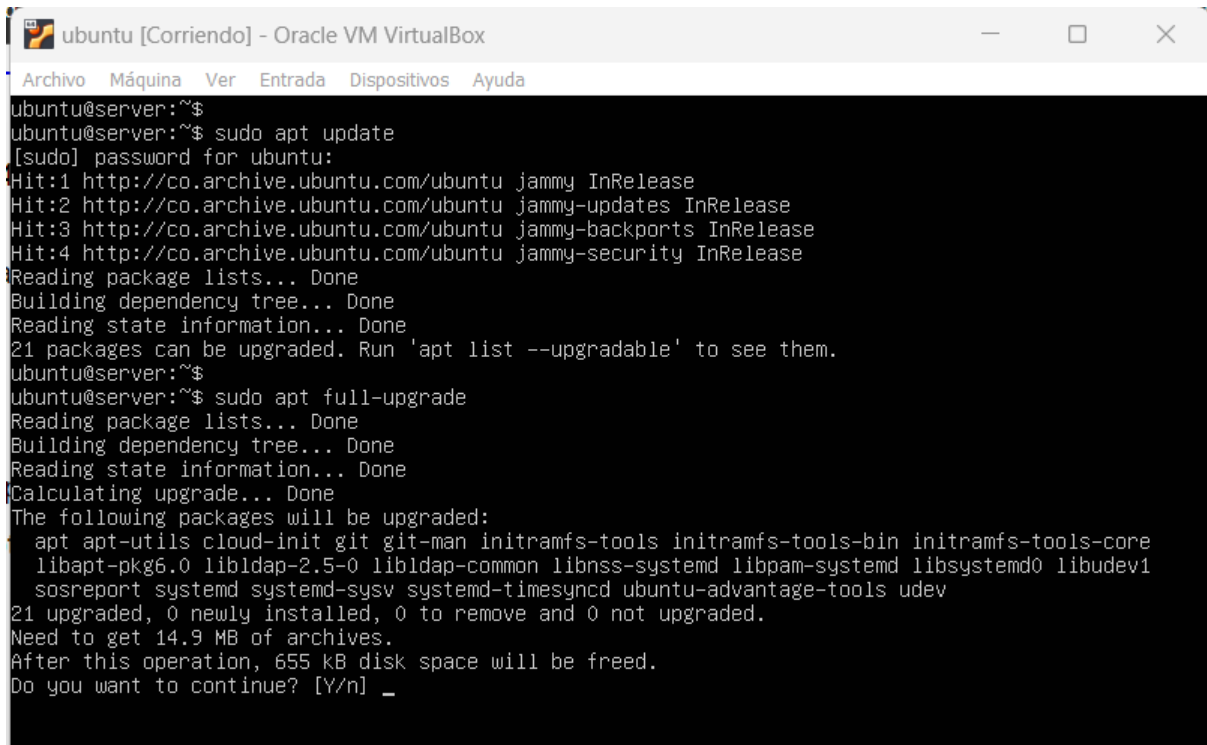
Para crear y usar volúmenes en Docker, puedes utilizar la línea de comandos de Docker o definir volúmenes en un archivo de configuración de Docker Compose. Algunos comandos comunes para trabajar con volúmenes incluyen `docker volume create`, `docker volume ls`, `docker volume inspect`, y más.

En resumen, los volúmenes en Docker son una herramienta importante para gestionar datos persistentes y compartir información entre contenedores, lo que facilita la implementación de aplicaciones en entornos de contenerización.

## BITÁCORA-CAPTURAS DE PANTALLA

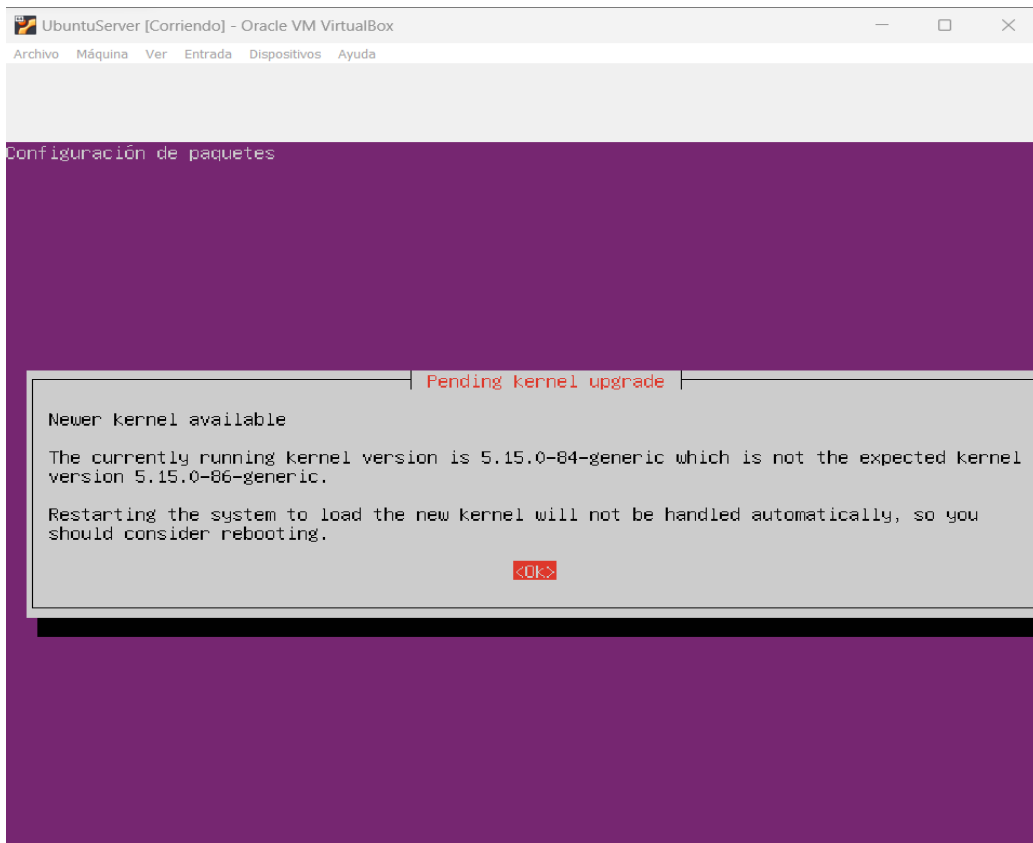
### Actualice la distribución

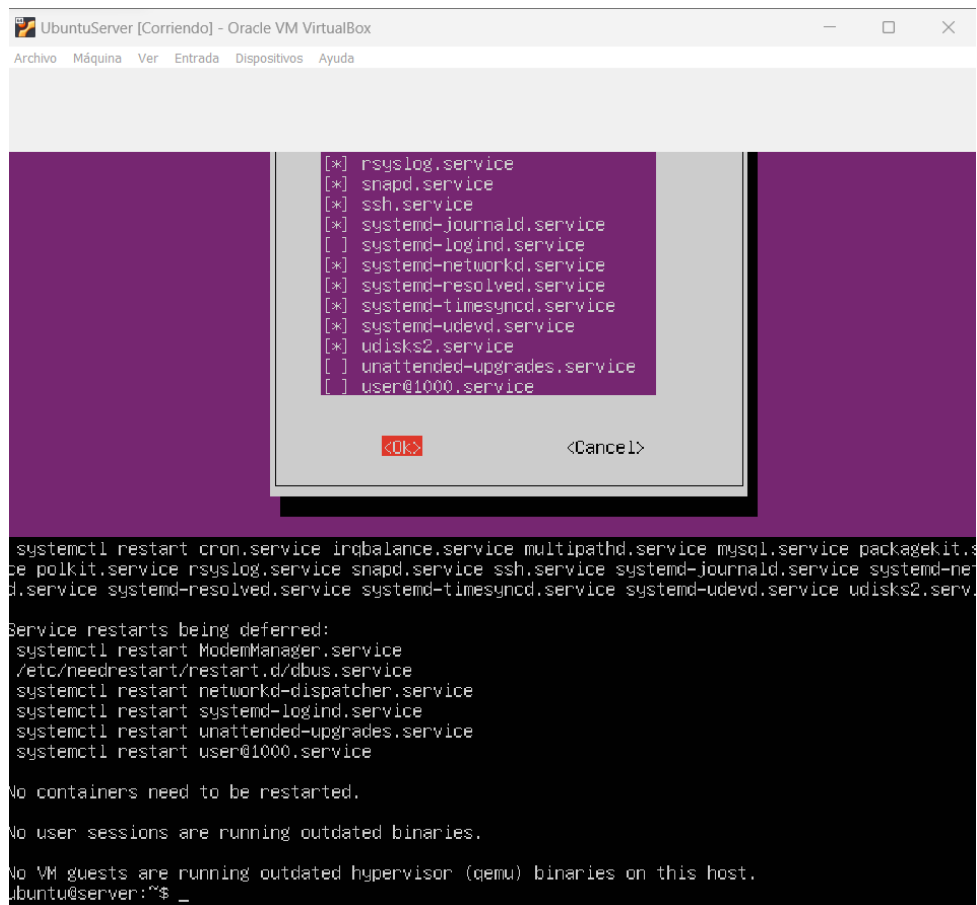
1)



A terminal window titled 'ubuntu [Corriendo] - Oracle VM VirtualBox'. The terminal shows the following commands and output:

```
ubuntu@server:~$  
ubuntu@server:~$ sudo apt update  
[sudo] password for ubuntu:  
Hit:1 http://co.archive.ubuntu.com/ubuntu jammy InRelease  
Hit:2 http://co.archive.ubuntu.com/ubuntu jammy-updates InRelease  
Hit:3 http://co.archive.ubuntu.com/ubuntu jammy-backports InRelease  
Hit:4 http://co.archive.ubuntu.com/ubuntu jammy-security InRelease  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
21 packages can be upgraded. Run 'apt list --upgradable' to see them.  
ubuntu@server:~$  
ubuntu@server:~$ sudo apt full-upgrade  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
Calculating upgrade... Done  
The following packages will be upgraded:  
  apt apt-utils cloud-init git git-man initramfs-tools initramfs-tools-bin initramfs-tools-core  
  libapt-pkg6.0 libldap-2.5-0 libldap-common libnss-systemd libpam-systemd libsystemd0 libudev1  
  sosreport systemd systemd-sysv systemd-timesyncd ubuntu-advantage-tools udev  
21 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.  
Need to get 14.9 MB of archives.  
After this operation, 655 kB disk space will be freed.  
Do you want to continue? [Y/n] _
```





```
systemctl restart cron.service irqbalance.service multipathd.service mysql.service packagekit.service polkit.service rsyslog.service snapd.service ssh.service systemd-journald.service systemd-networkd.service systemd-resolved.service systemd-timesyncd.service systemd-udev.service udisks2.service unattended-upgrades.service user@1000.service

Service restarts being deferred:
systemctl restart ModemManager.service
systemctl restart /etc/needrestart/restart.d/dbus.service
systemctl restart networkd-dispatcher.service
systemctl restart systemd-logind.service
systemctl restart unattended-upgrades.service
systemctl restart user@1000.service

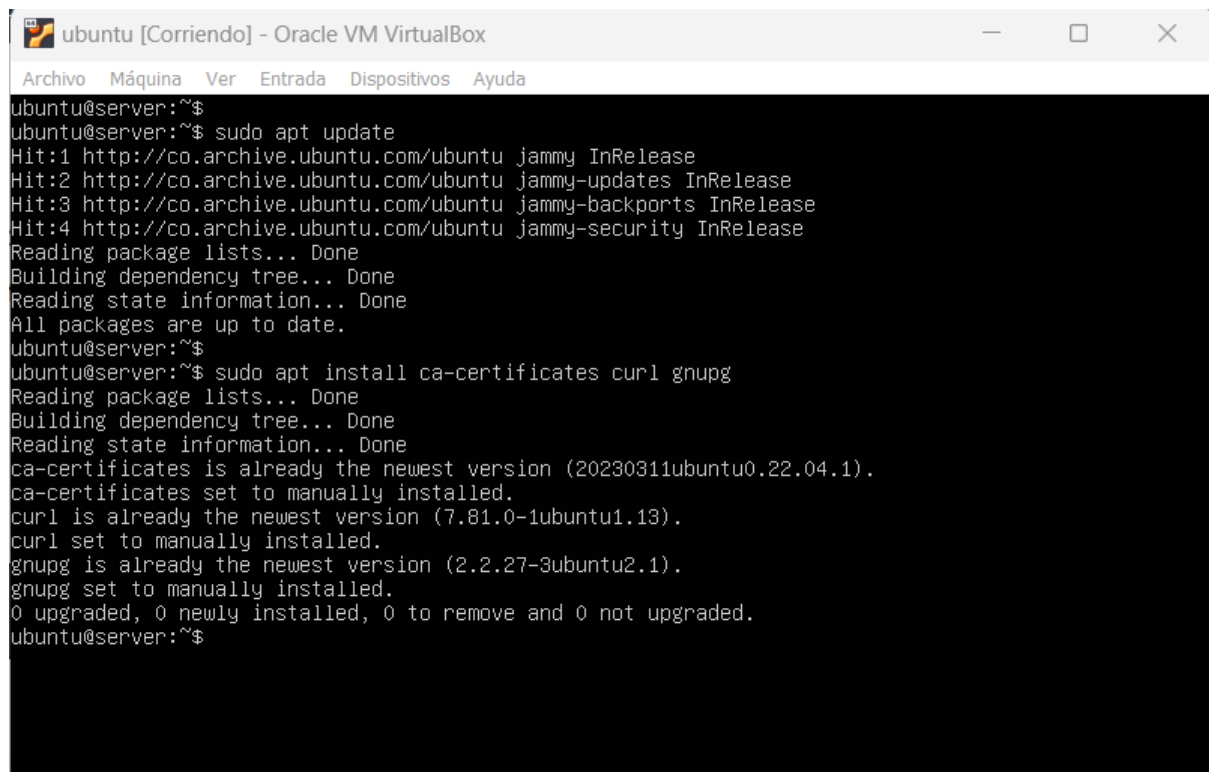
No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@server:~$ _
```

## Añada los repositorios de Docker

1)



```
ubuntu@server:~$
ubuntu@server:~$ sudo apt update
Hit:1 http://co.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://co.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:3 http://co.archive.ubuntu.com/ubuntu jammy-backports InRelease
Hit:4 http://co.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.
ubuntu@server:~$
ubuntu@server:~$ sudo apt install ca-certificates curl gnupg
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ca-certificates is already the newest version (20230311ubuntu0.22.04.1).
ca-certificates set to manually installed.
curl is already the newest version (7.81.0-1ubuntu1.13).
curl set to manually installed.
gnupg is already the newest version (2.2.27-3ubuntu2.1).
gnupg set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
ubuntu@server:~$
```

## 2) Añade la clave GPG oficial de Docker

```
ubuntu [Corriendo] - Oracle VM VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
ubuntu@server:~$
ubuntu@server:~$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
ubuntu@server:~$
```

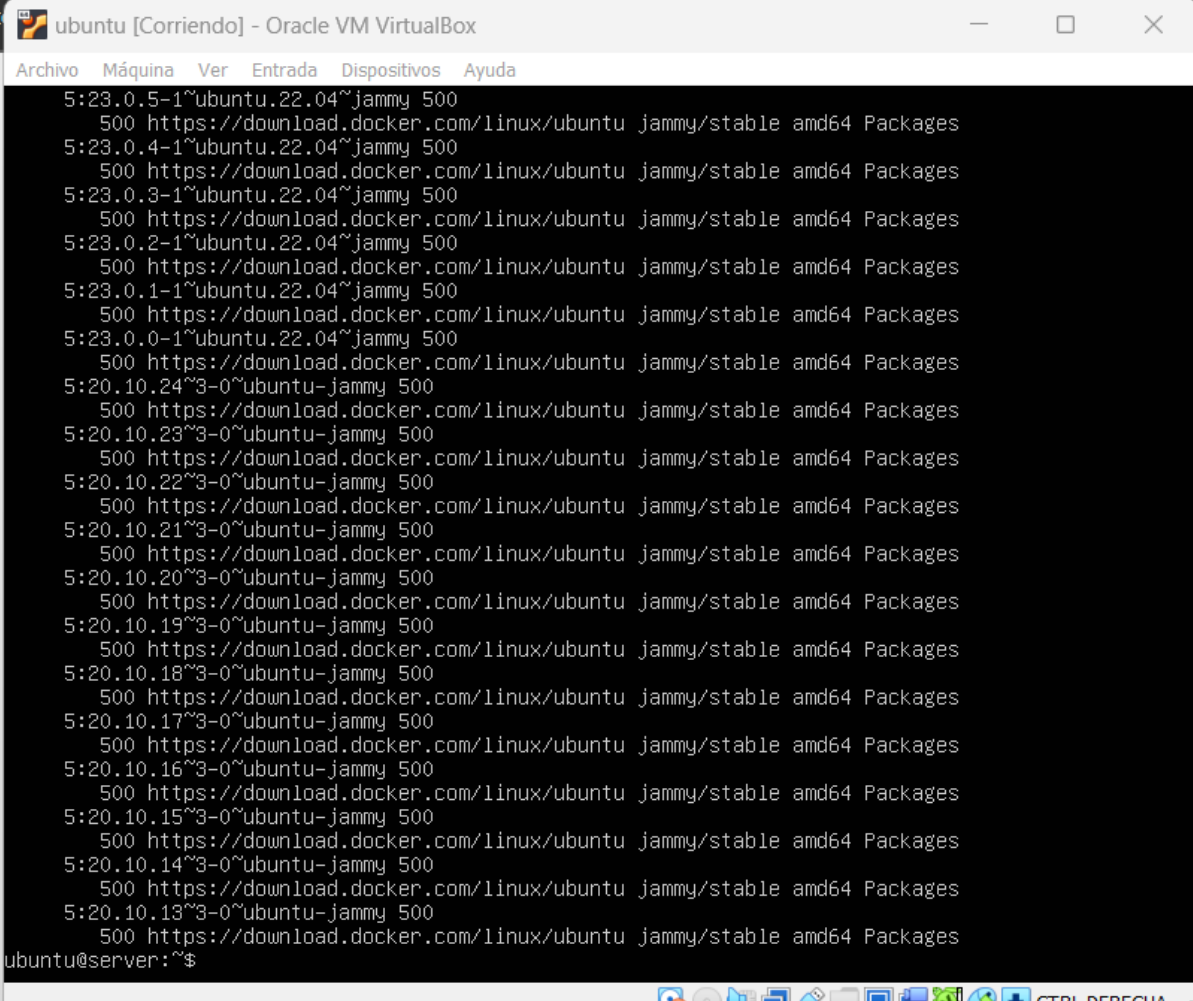
## 3) Añadir repositorio correspondiente a su versión de Ubuntu (escribiendo todo en una sola línea)

```
ubuntu [Corriendo] - Oracle VM VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
ubuntu@server:~$
ubuntu@server:~$
ubuntu@server:~$ sudo echo "deb [arch=\"$(dpkg --print-architecture)\" signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \"$(. /etc/os-release && echo \"$VERSION_CODENAME\")\" stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
ubuntu@server:~$
ubuntu@server:~$ _
```

## 4) Actualizar APT

```
ubuntu [Corriendo] - Oracle VM VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
ubuntu@server:~$
ubuntu@server:~$ sudo apt update
Hit:1 http://co.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://co.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:3 http://co.archive.ubuntu.com/ubuntu jammy-backports InRelease
Hit:4 http://co.archive.ubuntu.com/ubuntu jammy-security InRelease
Hit:5 https://download.docker.com/linux/ubuntu jammy InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
All packages are up to date.
ubuntu@server:~$
ubuntu@server:~$
```

## 5) Comprobar que APT se conecta con el repositorio adecuado = apt policy docker-ce



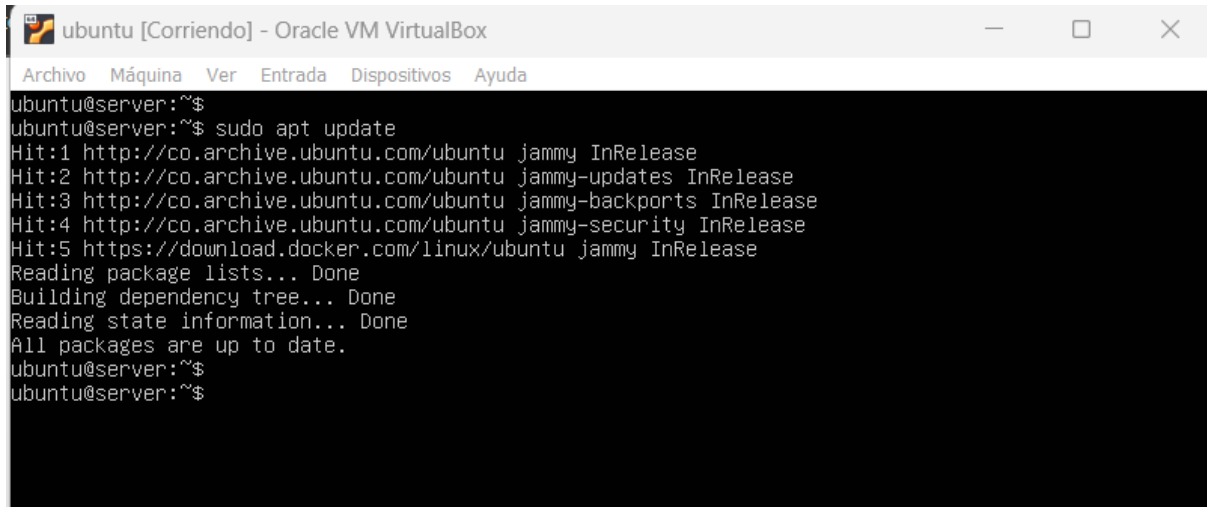
The screenshot shows a terminal window titled "ubuntu [Corriendo] - Oracle VM VirtualBox". The terminal output displays the APT policy for the 'docker-ce' package, showing its status as '500' and the repository it is configured to use: 'https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages'. The output is repeated for multiple instances of the package, corresponding to different IP addresses and versions. The terminal window includes a menu bar with options like 'Archivo', 'Máquina', 'Ver', 'Entrada', 'Dispositivos', and 'Ayuda'. The bottom of the window shows a taskbar with various system icons and a 'CTRL DERECHA' button.

```
ubuntu [Corriendo] - Oracle VM VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
5:23.0.5-1~ubuntu.22.04~jammy 500
500 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages
5:23.0.4-1~ubuntu.22.04~jammy 500
500 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages
5:23.0.3-1~ubuntu.22.04~jammy 500
500 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages
5:23.0.2-1~ubuntu.22.04~jammy 500
500 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages
5:23.0.1-1~ubuntu.22.04~jammy 500
500 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages
5:23.0.0-1~ubuntu.22.04~jammy 500
500 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages
5:20.10.24~3-0~ubuntu-jammy 500
500 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages
5:20.10.23~3-0~ubuntu-jammy 500
500 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages
5:20.10.22~3-0~ubuntu-jammy 500
500 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages
5:20.10.21~3-0~ubuntu-jammy 500
500 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages
5:20.10.20~3-0~ubuntu-jammy 500
500 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages
5:20.10.19~3-0~ubuntu-jammy 500
500 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages
5:20.10.18~3-0~ubuntu-jammy 500
500 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages
5:20.10.17~3-0~ubuntu-jammy 500
500 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages
5:20.10.16~3-0~ubuntu-jammy 500
500 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages
5:20.10.15~3-0~ubuntu-jammy 500
500 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages
5:20.10.14~3-0~ubuntu-jammy 500
500 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages
5:20.10.13~3-0~ubuntu-jammy 500
500 https://download.docker.com/linux/ubuntu jammy/stable amd64 Packages
ubuntu@server:~$
```



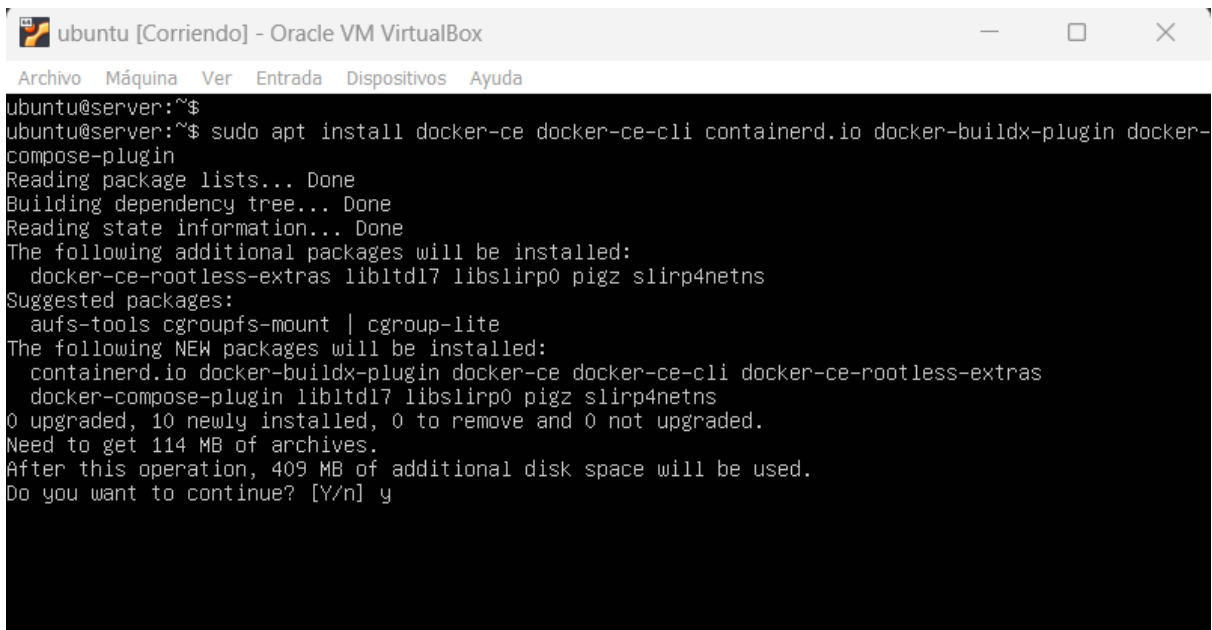
## INSTALE DOCKER

### 1) Actualizar administrador de paquetes **sudo apt update**

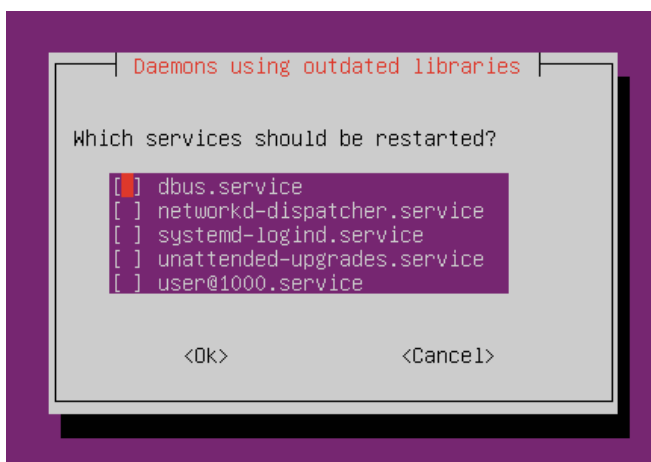


```
ubuntu [Corriendo] - Oracle VM VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
ubuntu@server:~$
ubuntu@server:~$ sudo apt update
Hit:1 http://co.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://co.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:3 http://co.archive.ubuntu.com/ubuntu jammy-backports InRelease
Hit:4 http://co.archive.ubuntu.com/ubuntu jammy-security InRelease
Hit:5 https://download.docker.com/linux/ubuntu jammy InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
All packages are up to date.
ubuntu@server:~$
ubuntu@server:~$
```

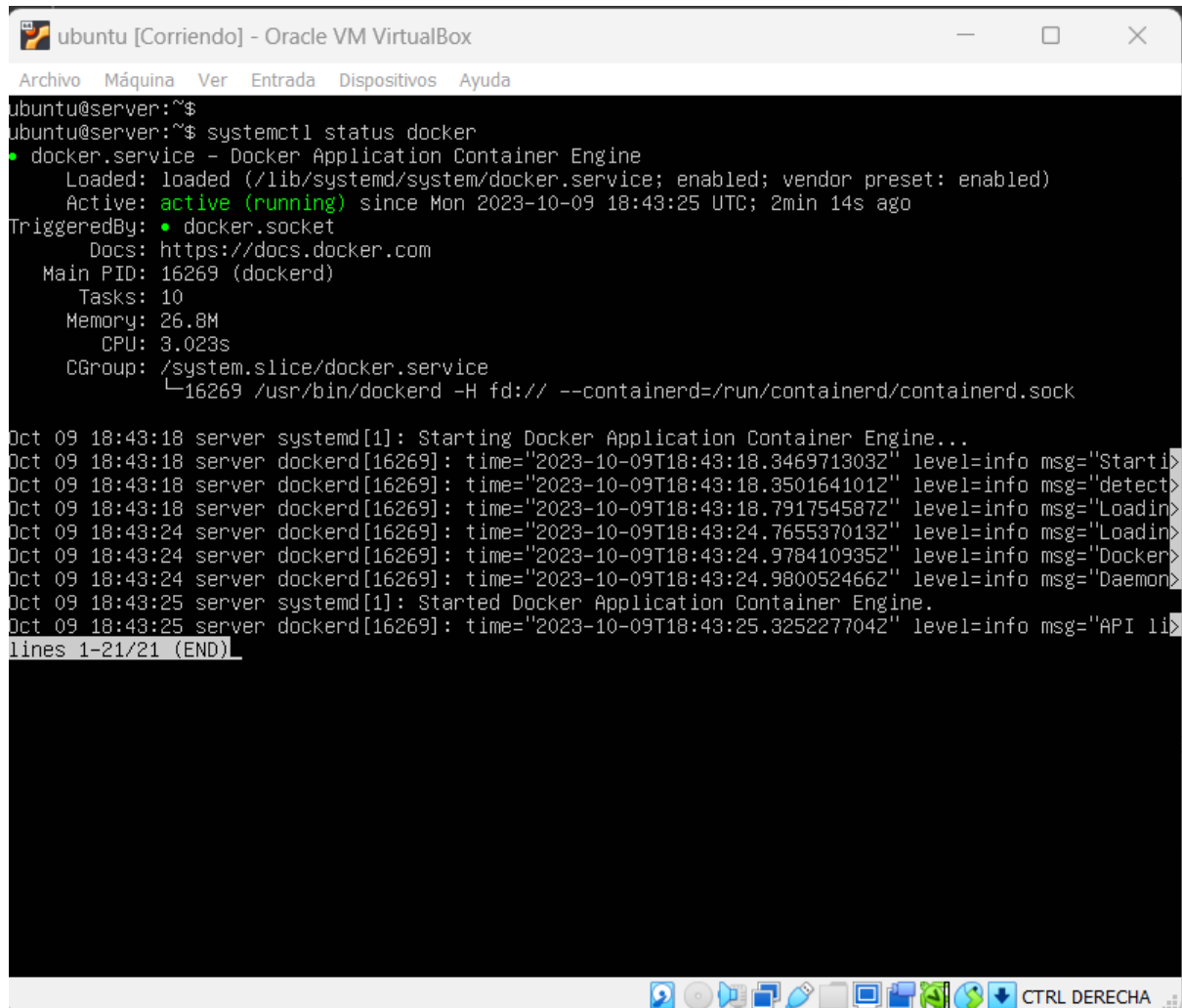
### 2) Instalar ultima version de Docker



```
ubuntu [Corriendo] - Oracle VM VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
ubuntu@server:~$
ubuntu@server:~$ sudo apt 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
The following additional packages will be installed:
  docker-ce-rootless-extras libltdl7 libslirp0 pigz slirp4netns
Suggested packages:
  aufs-tools cgroupfs-mount | cgroup-lite
The following NEW packages will be installed:
  containerd.io docker-buildx-plugin docker-ce docker-ce-cli docker-ce-rootless-extras
  docker-compose-plugin libltdl7 libslirp0 pigz slirp4netns
0 upgraded, 10 newly installed, 0 to remove and 0 not upgraded.
Need to get 114 MB of archives.
After this operation, 409 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
```

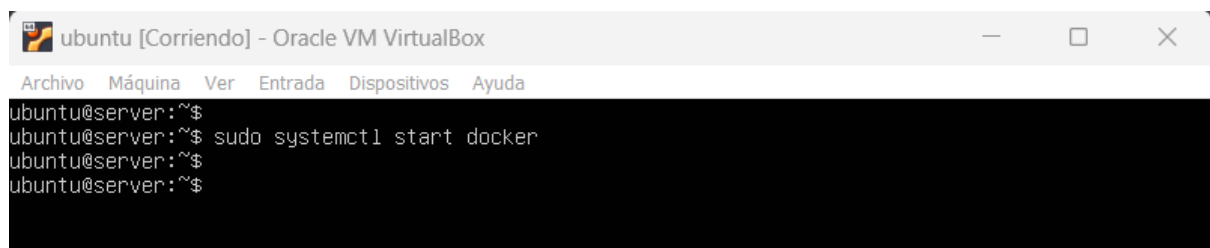


### 3)Comprobar que el servicio Docker este activo con comando **systemctl status docker**



```
ubuntu@server:~$  
ubuntu@server:~$ systemctl status docker  
• docker.service - Docker Application Container Engine  
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)  
   Active: active (running) since Mon 2023-10-09 18:43:25 UTC; 2min 14s ago  
TriggeredBy: • docker.socket  
   Docs: https://docs.docker.com  
   Main PID: 16269 (dockerd)  
     Tasks: 10  
    Memory: 26.8M  
       CPU: 3.023s  
   CGroup: /system.slice/docker.service  
           └─16269 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock  
  
Oct 09 18:43:18 server systemd[1]: Starting Docker Application Container Engine...  
Oct 09 18:43:18 server dockerd[16269]: time="2023-10-09T18:43:18.346971303Z" level=info msg="Starti>  
Oct 09 18:43:18 server dockerd[16269]: time="2023-10-09T18:43:18.350164101Z" level=info msg="detect>  
Oct 09 18:43:18 server dockerd[16269]: time="2023-10-09T18:43:18.791754587Z" level=info msg="Loadin>  
Oct 09 18:43:24 server dockerd[16269]: time="2023-10-09T18:43:24.765537013Z" level=info msg="Loadin>  
Oct 09 18:43:24 server dockerd[16269]: time="2023-10-09T18:43:24.978410935Z" level=info msg="Docker>  
Oct 09 18:43:24 server dockerd[16269]: time="2023-10-09T18:43:24.980052466Z" level=info msg="Daemon>  
Oct 09 18:43:25 server systemd[1]: Started Docker Application Container Engine.  
Oct 09 18:43:25 server dockerd[16269]: time="2023-10-09T18:43:25.325227704Z" level=info msg="API li>  
lines 1-21/21 (END)
```

En caso de que no estuviera activo utilizamos el comando **sudo systemctl start docker** para arrancar el servicio



```
ubuntu@server:~$  
ubuntu@server:~$ sudo systemctl start docker  
ubuntu@server:~$  
ubuntu@server:~$
```

## IMAGEN HELLO-WORLD

1) Iniciamos comprobando que no hay ningún contenedor creado (la opción -a hace que se muestren también los contenedores detenidos, sin ella se muestran sólo los contenedores que estén en marcha) **sudo docker ps -a**

```
ubuntu [Corriendo] - Oracle VM VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
ubuntu@server:~$
ubuntu@server:~$ sudo docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
ubuntu@server:~$
ubuntu@server:~$
```

2) Se comprueba que no se dispone de ninguna imagen **sudo docker image ls**

```
ubuntu [Corriendo] - Oracle VM VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
ubuntu@server:~$
ubuntu@server:~$ sudo docker image ls
REPOSITORY    TAG       IMAGE ID   CREATED   SIZE
ubuntu@server:~$
ubuntu@server:~$
```

3) Se crea contenedor con la aplicación de ejemplo hello-world **sudo docker run hello-world**

```
ubuntu [Corriendo] - Oracle VM VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
ubuntu@server:~$
ubuntu@server:~$ sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
719385e32844: Pull complete
Digest: sha256:4f53e2564790c8e7856ec08e384732aa38dc43c52f02952483e3f003afbf23db
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/

ubuntu@server:~$
```

#### 4) Listamos las imágenes existentes **sudo docker image ls**

```
ubuntu [Corriendo] - Oracle VM VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
ubuntu@server:~$
ubuntu@server:~$ sudo docker image ls
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
hello-world    latest    9c7a54a9a43c   5 months ago   13.3kB
ubuntu@server:~$
```

#### 5) Listamos contenedores existentes **sudo docker ps -a**

```
ubuntu [Corriendo] - Oracle VM VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
ubuntu@server:~$
ubuntu@server:~$ sudo docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
b9f093e10571   hello-world  "/hello"  4 minutes ago  Exited (0) 4 minutes ago           elastic_poitras
ubuntu@server:~$
```

6) Se pueden crear múltiples contenedores a partir de una imagen. Una vez que la imagen está disponible localmente, Docker no necesita volver a descargarla, lo que hace que la creación de contenedores sea instantánea. Aunque la descarga inicial puede llevar tiempo en imágenes más grandes.

Usar la opción `-d` se recomienda generalmente, ya que inicia el contenedor en segundo plano (modo detached) y permite mantener acceso a la terminal. Aunque con la imagen "hello-world" no es necesario, ya que se detiene automáticamente después de mostrar un mensaje. Cuando se crea un contenedor "hello-world" con la opción `-d`, no se muestra el mensaje en pantalla, simplemente se muestra el identificador completo del contenedor.

```
ubuntu [Corriendo] - Oracle VM VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
ubuntu@server:~$
ubuntu@server:~$ sudo docker run -d hello-world
e076f82e8cab06660d6a43ea0a8f8cce9733d90d5ee84257ea46bf0beb39840b
ubuntu@server:~$
ubuntu@server:~$ _
```

#### 7) Si listamos contenedores existentes:

```
ubuntu [Corriendo] - Oracle VM VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
ubuntu@server:~$
ubuntu@server:~$
ubuntu@server:~$ sudo docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
e076f82e8cab   hello-world  "/hello"  5 minutes ago  Exited (0) 5 minutes ago           less_lalande
b9f093e10571   hello-world  "/hello"  13 minutes ago  Exited (0) 13 minutes ago           elastic_poitras
ubuntu@server:~$
ubuntu@server:~$
```

8) Los contenedores se pueden destruir mediante el comando *rm*, haciendo referencia a ellos mediante su nombre o su id. No es necesario indicar el id completo, basta con escribir los primeros caracteres (de manera que no haya ambigüedades). Borre los dos contenedores existentes: **sudo docker rm**

```
ubuntu [Corriendo] - Oracle VM VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
ubuntu@server:~$
ubuntu@server:~$ sudo docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
e076f82e8cab   hello-world  "/hello"   6 minutes ago   Exited (0) 6 minutes ago           price
less_lalande
b9f093e10571   hello-world  "/hello"   14 minutes ago   Exited (0) 14 minutes ago           elast
ic_poitras
ubuntu@server:~$
ubuntu@server:~$
ubuntu@server:~$ sudo docker rm e07
e07
ubuntu@server:~$ sudo docker rm elastic_poitras
elastic_poitras
ubuntu@server:~$
ubuntu@server:~$ sudo docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
ubuntu@server:~$ _
```

9) Podemos dar nombre a los contenedores al ser creados

```
ubuntu [Corriendo] - Oracle VM VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
ubuntu@server:~$
ubuntu@server:~$ sudo docker run -d --name=hola-1 hello-world
066d4d0e9d21397db1e1d114a898a6f32e6bab17eb99d6a04e7abe88fb279c44
ubuntu@server:~$
ubuntu@server:~$ _
```

10) Listamos contenedores existentes

```
ubuntu [Corriendo] - Oracle VM VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
ubuntu@server:~$
ubuntu@server:~$ sudo docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
066d4d0e9d21   hello-world  "/hello"   About a minute ago   Exited (0) About a minute ago           hola-1
ubuntu@server:~$
```

11) Esto pasará si intentamos crear un contenedor con un nombre ya usado

```
ubuntu [Corriendo] - Oracle VM VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
ubuntu@server:~$
ubuntu@server:~$ sudo docker run -d --name=hola-1 hello-world
docker: Error response from daemon: Conflict. The container name "/hola-1" is already in use by container "066d4d0e9d21397db1e1d114a898a6f32e6bab17eb99d6a04e7abe88fb279c44". You have to remove (or rename) that container to be able to reuse that name.
See 'docker run --help'.
ubuntu@server:~$
ubuntu@server:~$ _
```

# IMAGEN APACHE

## Crear contenedor Apache

1) Se crea un contenedor que contenga un servidor Apache a partir de la imagen [bitnami/apache](https://hub.docker.com/r/bitnami/apache) la opción -P hace que Docker asigne de forma aleatoria un puerto de la máquina virtual al puerto asignado a Apache en el contenedor. La imagen bitnami/apache asigna a Apache el puerto 8080 del contenedor para conexiones http y el puerto 8443 para conexiones https.

```
ubuntu [Corriendo] - Oracle VM VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
ubuntu@server:~$
ubuntu@server:~$ sudo docker run -d -P --name=apache-1 bitnami/apache
Unable to find image 'bitnami/apache:latest' locally
latest: Pulling from bitnami/apache
8e5cf0abe9fb: Pull complete
Digest: sha256:19632a24f13ea4bc3ee551489f32c96c2fb752b177e15793a2214e124bf35210
Status: Downloaded newer image for bitnami/apache:latest
9723d9198d340d144d6b493f6e22f92f886dffc0aeabc920425667892b5b3c8
ubuntu@server:~$
ubuntu@server:~$
```

2) Consultar puerto del host utilizado por el contenedor

```
ubuntu [Corriendo] - Oracle VM VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
ubuntu@server:~$
ubuntu@server:~$ sudo docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        NAMES
9723d9198d34   bitnami/apache "/opt/bitnami/script..." 3 minutes ago  Up 3 minutes  apache-1
0.0.0.0:32769->8080/tcp, :::32769->8080/tcp, 0.0.0.0:32768->8443/tcp, :::32768->8443/tcp
066d4d0e9d21   hello-world    "/hello"                 9 minutes ago  Exited (0) 9 minutes ago  hola-1
ubuntu@server:~$ _
```

3) Podemos comprobar en el navegador la página inicial del contenedor y que se muestra una página que dice "It works!".



It works!

Para identificar mi ip use el siguiente comando **hostname -I**

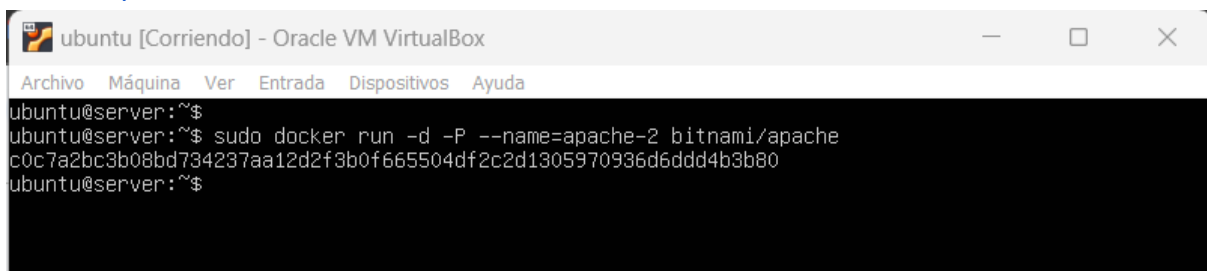
```
ubuntu [Corriendo] - Oracle VM VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
ubuntu@server:~$
ubuntu@server:~$ hostname -I
10.132.21.40 172.17.0.1
ubuntu@server:~$
```

## Modificar la página inicial del contenedor apache

En este apartado vamos a modificar la página web inicial de Apache del contenedor Docker.

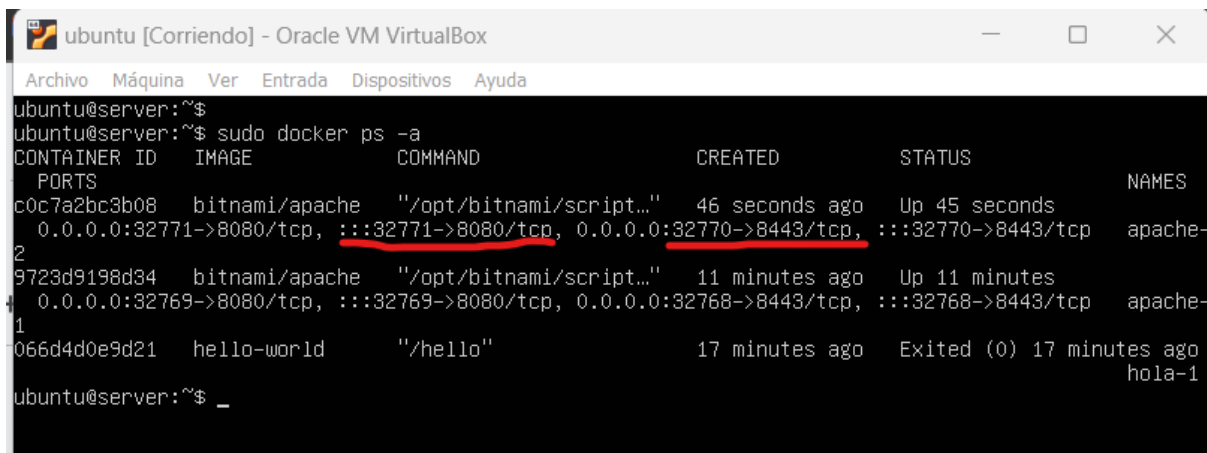
Tenga en cuenta que modificar el contenido de un contenedor tal y como vamos a hacer en este apartado sólo es aconsejable en un entorno de desarrollo, pero no es aconsejable en un entorno de producción porque va en contra de la "filosofía" de Docker. Los contenedores de Docker están pensados como objetos de "usar y tirar", es decir, para ser creados, destruidos y creados de nuevo tantas veces como sea necesario y en la cantidad que sea necesaria. En el apartado siguiente realizaremos la misma tarea de una forma más conveniente, modificando no el contenedor sino la imagen a partir de la cual se crean los contenedores.

1) Se crea un segundo contenedor que contenga un servidor Apache a partir de la imagen [bitnami/apache](https://hub.docker.com/r/bitnami/apache/).



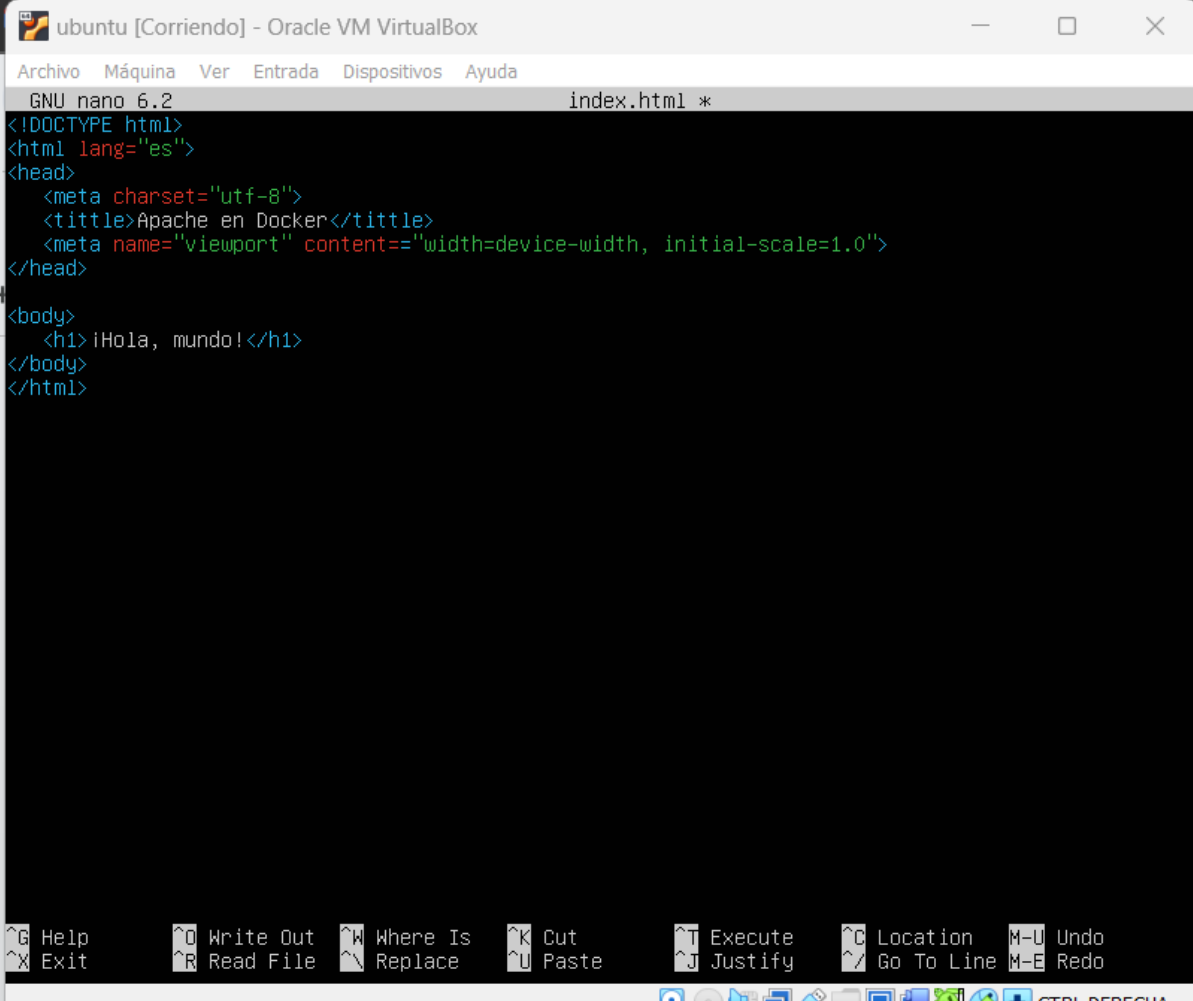
```
ubuntu [Corriendo] - Oracle VM VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
ubuntu@server:~$
ubuntu@server:~$ sudo docker run -d -P --name=apache-2 bitnami/apache
c0c7a2bc3b08bd734237aa12d2f3b0f665504df2c2d1305970936d6ddd4b3b80
ubuntu@server:~$
```

2) Se consulta puerto host utilizado por el contenedor



```
ubuntu [Corriendo] - Oracle VM VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
ubuntu@server:~$
ubuntu@server:~$ sudo docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS
c0c7a2bc3b08   bitnami/apache "/opt/bitnami/script..." 46 seconds ago Up 45 seconds 0.0.0.0:32771->8080/tcp, :::32771->8080/tcp, 0.0.0.0:32770->8443/tcp, :::32770->8443/tcp apache-2
9723d9198d34   bitnami/apache "/opt/bitnami/script..." 11 minutes ago Up 11 minutes 0.0.0.0:32769->8080/tcp, :::32769->8080/tcp, 0.0.0.0:32768->8443/tcp, :::32768->8443/tcp apache-1
066d4d0e9d21   hello-world    "/hello"                 17 minutes ago Exited (0) 17 minutes ago hola-1
ubuntu@server:~$ _
```

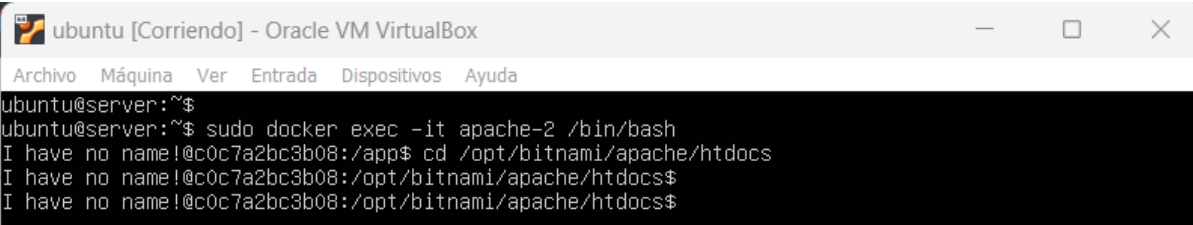
3) Se crea nueva página `index.html` **sudo nano index.html** en la cual escribiremos el siguiente contenido finalizando con el comando **CTRL + O** para guardar y **CTRL + X** para salir



The screenshot shows a terminal window titled "ubuntu [Corriendo] - Oracle VM VirtualBox". Inside, the GNU nano 6.2 editor is open with a file named "index.html \*". The code being entered is a basic HTML document. The bottom of the window shows a menu with various editing and execution commands like Help, Exit, Write Out, Read File, Where Is, Replace, Cut, Paste, Execute, Justify, Location, Go To Line, Undo, and Redo. A "CTRL DERECHA" button is also visible in the bottom right corner.

```
ubuntu [Corriendo] - Oracle VM VirtualBox
GNU nano 6.2 index.html *
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="utf-8">
  <title>Apache en Docker</title>
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body>
  <h1>¡Hola, mundo!</h1>
</body>
</html>
```

4) Se entra en la *shell* del contenedor para averiguar la ubicación de la página inicial. El DocumentRoot de Apache está en el directorio `/opt/bitnami/apache/htdocs`



The screenshot shows a terminal window with the following commands and output:

```
ubuntu@server:~$
ubuntu@server:~$ sudo docker exec -it apache-2 /bin/bash
I have no name!@c0c7a2bc3b08:/app$ cd /opt/bitnami/apache/htdocs
I have no name!@c0c7a2bc3b08:/opt/bitnami/apache/htdocs$
I have no name!@c0c7a2bc3b08:/opt/bitnami/apache/htdocs$
```

5) En ese directorio se encuentra el fichero `index.html` que queremos modificar:



The screenshot shows a terminal window with the following commands and output:

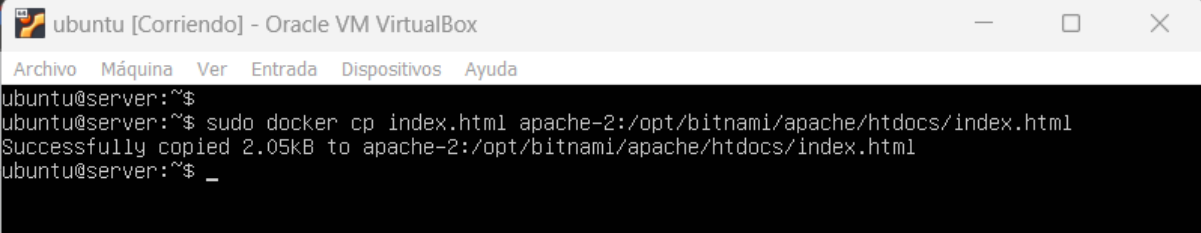
```
I have no name!@c0c7a2bc3b08:/opt/bitnami/apache/htdocs$
I have no name!@c0c7a2bc3b08:/opt/bitnami/apache/htdocs$ cat index.html
<html><body><h1>It works!</h1></body></html>
I have no name!@c0c7a2bc3b08:/opt/bitnami/apache/htdocs$
I have no name!@c0c7a2bc3b08:/opt/bitnami/apache/htdocs$
```



## 6)Salir de la shell **exit**

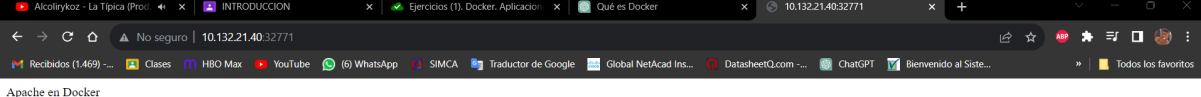
```
I have no name!@c0c7a2bc3b08:/opt/bitnami/apache/htdocs$  
I have no name!@c0c7a2bc3b08:/opt/bitnami/apache/htdocs$ | exit  
bash: syntax error near unexpected token `|'  
I have no name!@c0c7a2bc3b08:/opt/bitnami/apache/htdocs$ exit  
exit  
ubuntu@server:~$
```

## 7)Copie el fichero index.html en el contenedor



```
ubuntu@server:~$  
ubuntu@server:~$ sudo docker cp index.html apache-2:/opt/bitnami/apache/htdocs/index.html  
Successfully copied 2.05kB to apache-2:/opt/bitnami/apache/htdocs/index.html  
ubuntu@server:~$ _
```

## 8)Comprobar en navegador



Apache en Docker

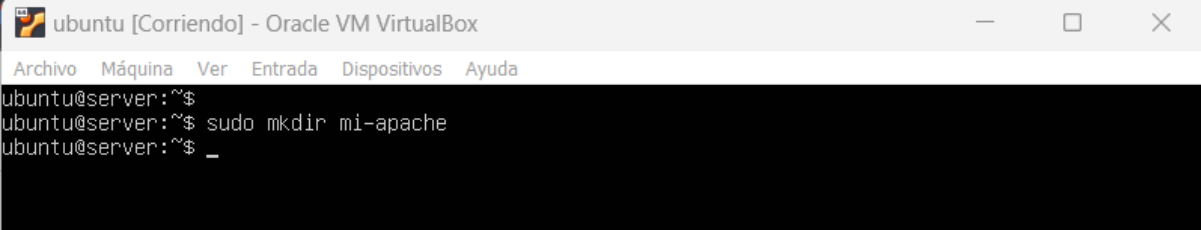
¡Hola, mundo!

## Crear nueva imagen

**Nota:** Si queremos cambiar la página inicial, la forma correcta de hacerlo en Docker es crear una nueva imagen que incluya la página modificada, de manera que cada vez que se cree el contenedor, la página inicial sea la modificada.

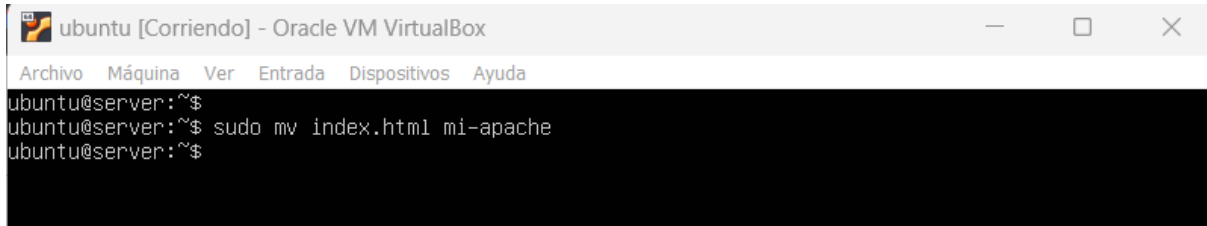
Las imágenes se crean a partir de Dockerfiles, ficheros que describen los elementos que forman la imagen. Los Dockerfiles pueden ser muy extensos. En este caso, se trata de un Dockerfile mínimo.

### 1)Cree un directorio que contendrá el Dockerfile **sudo mkdir mi-apache**



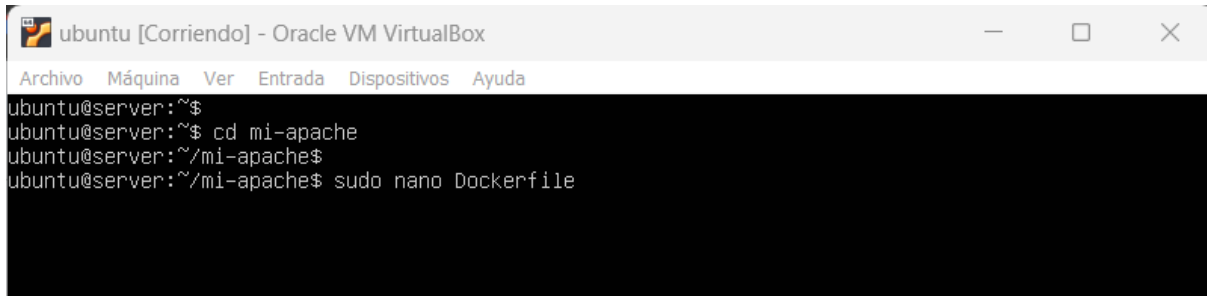
```
ubuntu@server:~$  
ubuntu@server:~$ sudo mkdir mi-apache  
ubuntu@server:~$ _
```

2) Se copia el fichero index.html creado anteriormente **sudo mv index.html mi-apache**

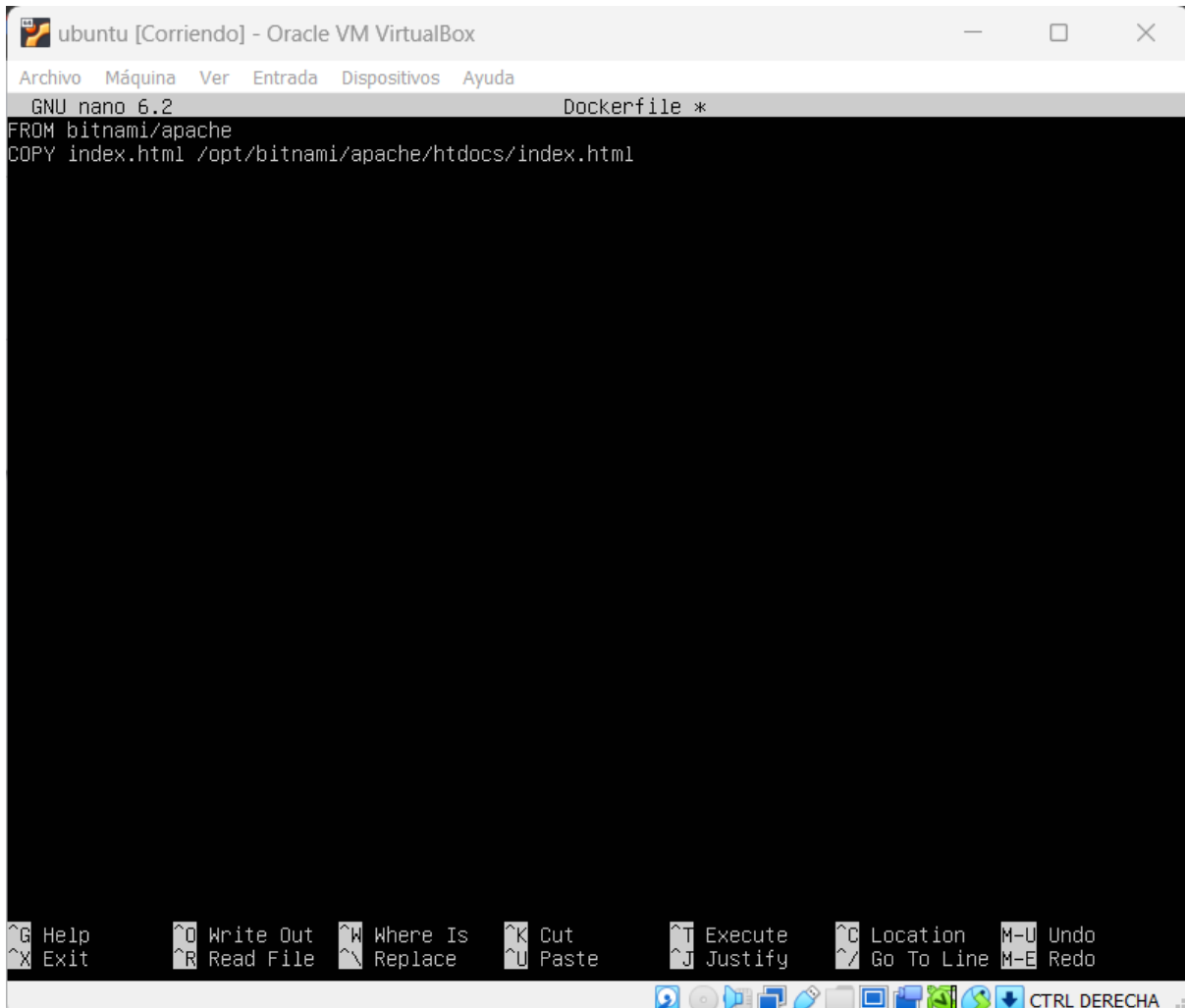


```
ubuntu [Corriendo] - Oracle VM VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
ubuntu@server:~$
ubuntu@server:~$ sudo mv index.html mi-apache
ubuntu@server:~$
```

3) Se ingresa al directorio y se crea un fichero Dockerfile



```
ubuntu [Corriendo] - Oracle VM VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
ubuntu@server:~$
ubuntu@server:~$ cd mi-apache
ubuntu@server:~/mi-apache$
ubuntu@server:~/mi-apache$ sudo nano Dockerfile
```



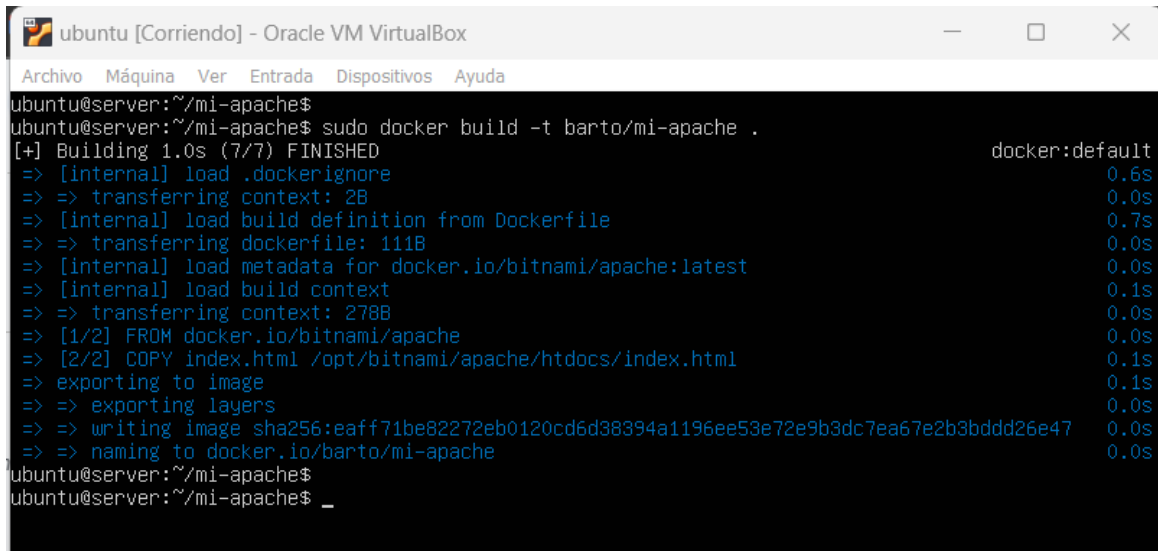
```
ubuntu [Corriendo] - Oracle VM VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
GNU nano 6.2 Dockerfile *
FROM bitnami/apache
COPY index.html /opt/bitnami/apache/htdocs/index.html
```

^G Help    ^O Write Out    ^W Where Is    ^K Cut    ^T Execute    ^C Location    M-U Undo  
^X Exit    ^R Read File    ^\_ Replace    ^U Paste    ^J Justify    ^\_ Go To Line    M-E Redo

CTRL DERECHA

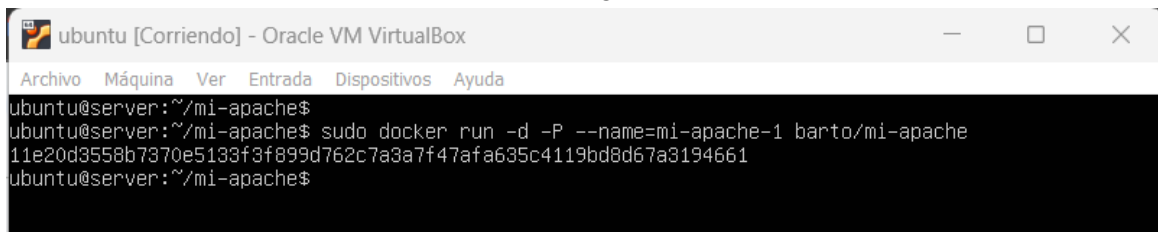
4) Genere la nueva imagen. El último argumento (y el único imprescindible) es el nombre del archivo Dockerfile que tiene que utilizar para generar la imagen. Como en este caso se encuentra en el mismo directorio y tiene el nombre predeterminado **Dockerfile**, se puede escribir simplemente punto (.).

Para indicar el nombre de la imagen se debe añadir la opción -t. El nombre de la imagen debe seguir el patrón **nombre-de-usuario/nombre-de-imagen**. Si la imagen sólo se va a utilizar localmente, el nombre de usuario y de la imagen pueden ser cualquier palabra.



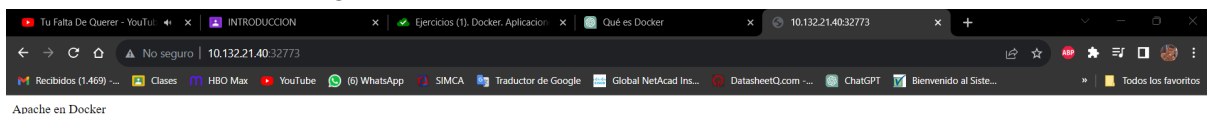
```
ubuntu [Corriendo] - Oracle VM VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
ubuntu@server:~/mi-apache$
ubuntu@server:~/mi-apache$ sudo docker build -t barto/mi-apache .
[+] Building 1.0s (7/7) FINISHED
=> [internal] load .dockerignore                                0.6s
=> => transferring context: 2B                                    0.0s
=> [internal] load build definition from Dockerfile              0.7s
=> => transferring dockerfile: 111B                               0.0s
=> [internal] load metadata for docker.io/bitnami/apache:latest  0.0s
=> [internal] load build context                                0.1s
=> => transferring context: 278B                                    0.0s
=> [1/2] FROM docker.io/bitnami/apache                          0.0s
=> [2/2] COPY index.html /opt/bitnami/apache/htdocs/index.html  0.1s
=> exporting to image                                           0.1s
=> => exporting layers                                           0.0s
=> => writing image sha256:eaff71be82272eb0120cd6d38394a1196ee53e72e9b3dc7ea67e2b3bddd26e47 0.0s
=> => naming to docker.io/barto/mi-apache                       0.0s
ubuntu@server:~/mi-apache$
ubuntu@server:~/mi-apache$ _
```

5) Se crea contenedor a partir de la nueva imagen



```
ubuntu [Corriendo] - Oracle VM VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
ubuntu@server:~/mi-apache$
ubuntu@server:~/mi-apache$ sudo docker run -d -P --name=mi-apache-1 barto/mi-apache
11e20d3558b7370e5133f3f899d762c7a3a7f47afa635c4119bd8d67a3194661
ubuntu@server:~/mi-apache$
```

6) Se comprueba en navegador



¡Hola, mundo!

## VOLÚMENES

Docker simplifica enormemente la creación de contenedores, y eso lleva a tratar los contenedores como un elemento efímero, que se crea cuando se necesita y que no importa que se destruya puesto que puede ser reconstruido una y otra vez a partir de su imagen.

Pero si la aplicación o aplicaciones incluidas en el contenedor generan datos y esos datos se guardan en el propio contenedor, en el momento en que se destruyera el contenedor perderíamos esos datos. Para conseguir la persistencia de los datos, se pueden emplear dos técnicas:

- Los directorios enlazados, en la que la información se guarda fuera de Docker, en la máquina *host* (en nuestro caso, en la máquina virtual de Ubuntu)
- Los volúmenes, en la que la información se guarda mediante Docker, pero en unos elementos llamados *volúmenes*, independientes de las imágenes y los contenedores

Los volúmenes son la mejor solución cuando la información es generada por el propio contenedor y los directorios enlazados pueden ser más adecuados cuando la información no es generada por ningún contenedor.

### Directorios enlazados (*bind*)

Docker permite asociar directorios del contenedor a directorios de la máquina *host* (en nuestro caso, de la máquina virtual de Ubuntu). Es decir, que cuando el contenedor lea o escriba en su directorio, donde leerá o escribirá será en el directorio de la máquina virtual.

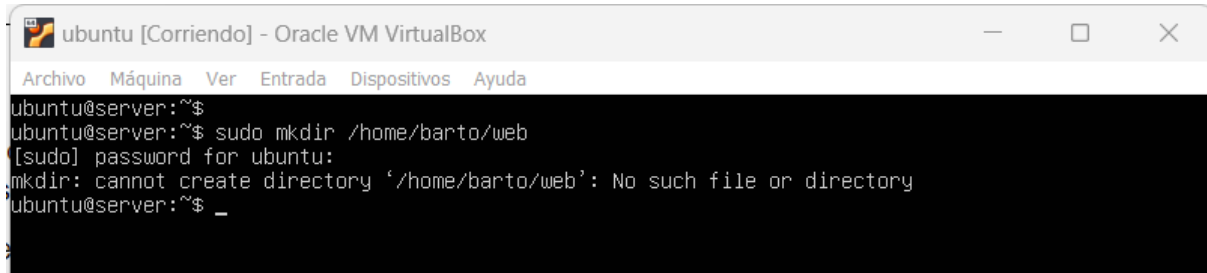
Si el directorio enlazado es el directorio en el que la aplicación guarda los datos generados por la propia aplicación, de esta manera conseguimos que los datos estén realmente fuera del contenedor. Eso significa que podemos conservar los datos aunque se destruya el contenedor, reutilizarlos con otro contenedor, etc.

En este ejemplo, vamos a volver a aprovechar el hecho que la imagen `bitnami/apache` está configurada para que el directorio `htdocs` habitual (en el caso de `bitnami` como hemos visto en el ejercicio anterior es `/opt/bitnami/apache/htdocs`) enlace al directorio `/app` [véase [Variables de entorno de bitnami/apache](#)]

Si al crear el contenedor enlazamos el directorio `/app` con un directorio de la máquina virtual, el contenedor servirá las páginas contenidas en el directorio de la máquina virtual.

1) Se crea un directorio que contendrá las páginas web

Hasta este punto fue posible realizar la práctica, debido a que los comandos ejecutados no daban con el resultado deseado, desde este punto al querer crear un directorio no lo permite, como se muestra en la imagen

A screenshot of a terminal window titled 'ubuntu [Corriendo] - Oracle VM VirtualBox'. The window has a menu bar with 'Archivo', 'Máquina', 'Ver', 'Entrada', 'Dispositivos', and 'Ayuda'. The terminal shows the following commands and output:

```
ubuntu@server:~$  
ubuntu@server:~$ sudo mkdir /home/barto/web  
[sudo] password for ubuntu:  
mkdir: cannot create directory '/home/barto/web': No such file or directory  
ubuntu@server:~$ _
```

## CONCLUSIÓN

En conclusión, Docker ha cambiado la forma en que desarrollamos y ejecutamos aplicaciones, ofreciendo un enfoque más ágil y eficiente. A medida que continúa evolucionando y madurando, es esencial que las organizaciones comprendan sus ventajas y desafíos, y tomen decisiones informadas sobre su adopción. En última instancia, Docker es una herramienta poderosa que puede impulsar la innovación y la eficiencia en el desarrollo y despliegue de software.