Introducción a Docker Containers

28 de julio de 2025

¿Qué es Docker?

- Docker es un proyecto de código abierto pero de una compañia privada que **automatiza el despliegue de aplicaciones de software dentro de contenedores**.
- Proporciona una capa adicional de abstracción de **virtualización a nivel de SO** en Linux.
- En palabras más sencillas, Docker es una herramienta que permite a desarrolladores, administradores de sistemas, etc., **desplegar fácilmente sus aplicaciones en un entorno aislado (llamado contenedores)** para que se ejecuten en el sistema operativo anfitrión, es decir, Linux.
- El beneficio clave de Docker es que permite a los usuarios **empaquetar una aplicación con todas sus dependencias en una unidad estandarizada** para el desarrollo de software.

¿Qué son los Contenedores? (*Containers*)

- Antes se usaban mucho las Máquinas Virtuales.
- Las VMs ejecutan aplicaciones dentro de un **sistema operativo invitado** que se ejecuta en hardware virtualizado, lo que conlleva una sobrecarga computacional sustancial.
- Los **contenedores** son diferentes: usan los mecanismos de bajo nivel del sistema operativo anfitrión para proporcionar la mayor parte del aislamiento de las máquinas virtuales, pero **con consumen menos recursos**.
- Ofrecen un **mecanismo de empaquetado lógico** en el que las aplicaciones pueden ser abstraídas del entorno en el que realmente se ejecutan.
- Este desacoplamiento permite que las **aplicaciones basadas en contenedores sean portatiles**.
- Esto permite crear **entornos predecibles que están aislados** del resto de las aplicaciones y pueden ejecutarse en cualquier lugar.
- También ofrecen control más granular de los recursos.

Terminología Básica de Docker

La jerga específica de Docker puede ser confusa al principio. Aquí algunos términos clave:

- **Imágenes (Images)**: Son los planos de nuestra aplicación que forman la base de los contenedores. Se descargan del Docker registry (ej. Docker Hub) usando `docker pull`.
- **Contenedores (Containers)**: Se crean a partir de imágenes Docker y ejecutan la aplicación real. Se inician usando `docker run`.
- **Docker Daemon**: El servicio en segundo plano que se ejecuta en el anfitrión y gestiona la construcción, ejecución y distribución de contenedores Docker. Es el proceso con el que interactúan los clientes.
- **Docker Client**: La herramienta de línea de comandos que permite al usuario interactuar con el daemon. También pueden existir otros clientes con GUI.
- **Docker Hub**: Un registro de imágenes Docker. Puedes pensarlo como un directorio de todas las imágenes Docker disponibles.

Instalación de Docker

- Docker (la compañia) ha invertido significativamente en mejorar la experiencia de incorporación para sus usuarios en sistemas operativos como Mac, Linux y Windows, haciendo que su ejecución sea muy sencilla.
- Después de la instalación, puedes verificar que Docker funciona correctamente ejecutando un comando simple:

  ```
  $ docker run hello-world
  ```

- Esto debería mostrar un mensaje de "Hello from Docker", confirmando que tu instalación es correcta.

Uso Básico: Imágenes

- Para obtener una nueva imagen Docker, puedes descargarla de un registro (como Docker Hub) o crear la tuya propia.
- Existen decenas de miles de imágenes disponibles en Docker Hub.
- Use el comando `docker pull` para descargar una imagen de un registro a tu sistema.
- Para ver una lista de todas las imágenes disponibles localmente en tu sistema, usa el comando `docker images`.

Ejemplo: Descargar la imagen `busybox`

```
$ docker pull busybox
# Dependiendo de tu instalación, podrías necesitar 'sudo' en Linux.
```

Uso Básico: Imágenes

Ejemplo: Listar imágenes locales

```
$ docker images
REPOSITORY    TAG       IMAGE ID       CREATED         VIRTUAL SIZE
busybox       latest    c51f86c28340   4 weeks ago     1.109 MB
```

El `TAG` se refiere a una instantánea particular de la imagen y el `IMAGE ID` es su identificador único.

Uso Básico: Contenedores (Parte 1)

- Para ejecutar un contenedor basado en una imagen, se utiliza el comando `docker run`.

Ejemplo: Ejecutar `busybox` sin comando explícito

```
$ docker run busybox
$ # ¡No pasó nada visible! El contenedor se inició, ejecutó un comando vacío y
↪  salió.
```

Ejemplo: Ejecutar `busybox` con un comando

```
$ docker run busybox echo "hello from busybox"
hello from busybox
```

- El cliente de Docker ejecutó diligentemente el comando `echo` en el contenedor `busybox` y luego salió.
- Note que los contenedores son mucho más rápidos que las maquinas virtuales.

Uso Básico: Contenedores (Parte 2)

- El comando `docker ps` muestra todos los contenedores que se están ejecutando actualmente.
- Para ver una lista de todos los contenedores que se han ejecutado (incluyendo los que ya salieron), usa `docker ps -a`.

Ejemplo: `docker ps -a`

```
$ docker ps -a
CONTAINER ID   IMAGE         COMMAND     CREATED         STATUS
↪  PORTS    NAMES
305297d7a235   busybox       "uptime"    11 minutes ago  Exited (0) 11 minutes
↪  ago distracted_goldstine
ff0a5c3750b9   busybox       "sh"        12 minutes ago  Exited (0) 12 minutes
↪  ago elated_ramanujan
14e5bd11d164   hello-world   "/hello"    2 minutes ago   Exited (0) 2 minutes ago
↪  thirsty_euclid
```

Uso Básico: Contenedores (Parte 2)

Ejemplo: Ejecutar un contenedor en modo interactivo

- Para ejecutar múltiples comandos en un contenedor, puedes usar las banderas `-it` con `docker run`.
- Esto te conecta a una TTY interactiva en el contenedor.

```
$ docker run -it busybox sh
/ # ls
bin dev etc home proc root sys tmp usr var
/ # uptime
05:45:21 up 5:58, 0 users, load average: 0.00, 0.01, 0.04
/ # exit
```

Uso Básico: Eliminación de Contenedores e Imágenes

- Dejar contenedores inactivos consume espacio en disco, por lo que se recomienda limpiarlos una vez que hayas terminado con ellos.
- Puedes eliminar contenedores usando `docker rm` y el ID del contenedor.

Ejemplo: Eliminar contenedores por ID

```
$ docker rm 305297d7a235 ff0a5c3750b9
305297d7a235
ff0a5c3750b9
```

Ejemplo: Eliminar todos los contenedores con estado 'exited'

```
$ docker container prune
```

Ejemplo: Eliminar imágenes que ya no necesitas

```
$ docker rmi yourusername/catnip
```

Ejecutando un Sitio Web Estático con Docker

Es posible descargar y ejecutar una imagen directamente en una sola vez usando `docker run`.
Ejemplo: Ejecutar un sitio web estático (imagen `prakhar1989/static-site`)

- `--rm` elimina automáticamente el contenedor al salir.
- `-it` especifica una terminal interactiva para facilitar la detención con Ctrl+C.

```
$ docker run --rm -it prakhar1989/static-site
```

Si la imagen no existe localmente, el cliente la descargará primero y luego la ejecutará.

Uso Básico de Contenedores

Ejemplo: Ejecutar en modo desacoplado y publicar puertos

- `-d` desacopla tu terminal.
- `-P` publica todos los puertos expuestos a puertos aleatorios del host.
- `--name` asigna un nombre al contenedor.

```
$ docker run -d -P --name static-site prakhar1989/static-site
e61d12292d69556eabe2a44c16cbd54486b2527e2ce4f95438e504afb7b02810
```

Verificar los puertos publicados:

```
$ docker port static-site
80/tcp -> 0.0.0.0:32769
443/tcp -> 0.0.0.0:32768
```

Ahora puedes abrir `http://localhost:32769` en tu navegador.

Uso Básico: Contenedores (Parte 2)

Detener un contenedor desacoplado:

```
$ docker stop static-site
static-site
```

Desplegar esto en un servidor real solo requeriría instalar Docker y ejecutar el comando anterior.