Capítulo 3
Los contenedores

1. Especificaciones

El capítulo Aplicación standalone reveló las deficiencias de una arquitectura monolítica. El capítulo Infraestructura y servicios básicos sentó las bases de una infraestructura tolerante a fallos capaz de alojar aplicaciones en condiciones mucho mejores. Ahora tenemos que ser capaces de desplegar la aplicación en los servidores de la forma más sencilla y rápida posible. Hay muchos pasos a seguir para instalarla, configurarla y ponerla en marcha. Es hora de completar las especificaciones:

- La aplicación debe ser fácil de instalar.
- La aplicación debe ser reutilizable.
- La aplicación se debe poder desplegar en varios servidores.
- El mismo servidor debe poder ejecutar varias instancias de la aplicación.
- El mantenimiento de la aplicación debe ser mínimo.
- La aplicación no debe ocupar todos los recursos del servidor.

Este último punto puede sorprender, pero también es un factor de disponibilidad y tolerancia a fallos. Un proceso que consume todos los recursos de la CPU impide que otras aplicaciones y servicios funcionen correctamente.

Si consume toda la memoria, corre el riesgo de que Linux mate el proceso, mediante el mecanismo OOMKiller (*Out-Of-Memory Killer*). No hay que olvidar que el papel de un sistema operativo es preservar a toda costa el funcionamiento global del sistema y conservar los recursos, aunque ello implique sacrificar determinados servicios y aplicaciones.

Se pueden considerar varias soluciones clásicas. Todas las distribuciones Linux distribuyen componentes de software en forma de paquetes. Podríamos imaginar un paquete dpkg o rpm para nuestra aplicación, lo que facilitaría su instalación y el paquete sería reutilizable. Pero no se cumplirían totalmente las especificaciones: ¿cómo instalarlo varias veces en el mismo servidor? ¿Cómo evitar que ponga en peligro el servidor en su conjunto?

Da la casualidad de que el kernel de Linux dispone desde hace años de todo lo necesario para resolver el problema: mecanismos de aislamiento y contenedores

2. Aislamiento y contenedores

2.1 Principio

El aislamiento es una técnica que consiste en ejecutar aplicaciones en sus propios contextos, aisladas de las demás. Es una forma de virtualización. La primera implementación de este tipo en Unix se remonta a 1979, con **chroot**.

El núcleo de Linux utiliza dos mecanismos para aislar uno o más procesos: **namespaces** y **cgroups**. Los namespaces se introdujeron en 2002, y los cgroups en 2007. Todos los núcleos Linux incluyen estas características por defecto. La llegada de los **namespaces** a los usuarios en Kernel 3.8 en febrero de 2013 allanó el camino para el soporte completo de contenedores.

Los namespaces agrupan procesos en un espacio aislado de los demás. Se pueden utilizar para:

– un aislamiento de procesos y un proceso con PID 1 como primer proceso en su espacio (PID 1 del primer proceso en un espacio de nombres),
– un aislamiento de la red, con direcciones IP y puertos propios,
– un aislamiento de volúmenes de datos: almacenamiento,
– un aislamiento de permisos y usuarios en relación con el host: ser root dentro de un contenedor puede corresponder así a un usuario sin permiso sobre el host;

Los cgroups controlan los recursos del sistema que un grupo de procesos puede utilizar, con los siguientes controles, por ejemplo:

– limitar el consumo de memoria,
– limitar el uso de procesadores,
– gestionar prioridades,
– obtener información "contable" sobre este grupo,
– controlar varios procesos en su conjunto (detenerlos, por ejemplo),
– aislar este grupo asociándolo a un espacio de nombres.

La combinación de estos dos mecanismos es la base del principio del contenedor.

2.2 Contenedor y máquina virtual

Los contenedores se comparan a menudo con las máquinas virtuales, poder facilitar la comprensión, pero se trata de dos tipos de tecnología de virtualización bien distintas. El concepto es diferente.

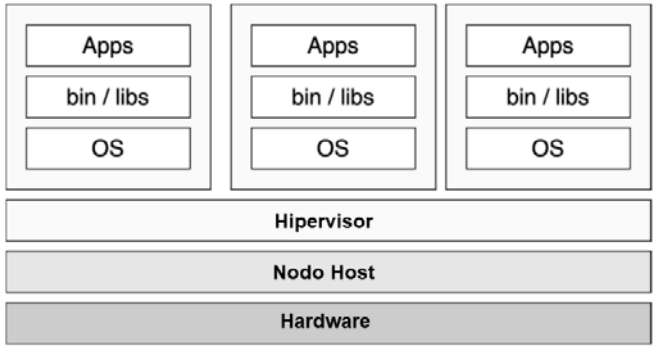

Figura 1: Hipervisor y máquinas virtuales

Las máquinas virtuales se ejecutan en hipervisores. Tanto si utilizan emulación como si no, las máquinas virtuales ejecutan un sistema operativo completo, con un núcleo y controladores de dispositivos (incluso si utilizan paravirtualización). Las aplicaciones se instalan en el sistema operativo de la máquina virtual. Desde el punto de vista del hipervisor, la máquina virtual es un único proceso.

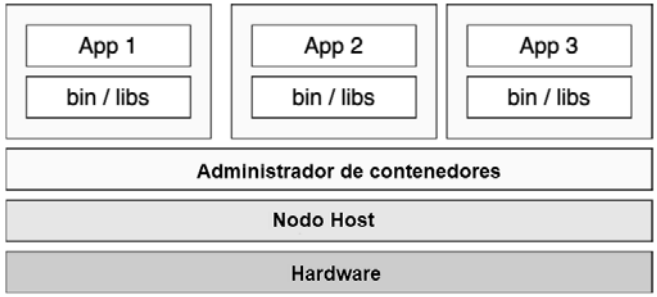

Figura 2: Host y contenedores

Los contenedores se ejecutan directamente en el host, en el mismo kernel. Los procesos que contienen son procesos como cualquier otro, sólo que aislados y limitados en recursos. Si un contenedor contiene diez procesos, puede ver diez procesos en su host.

2.3 Namespace

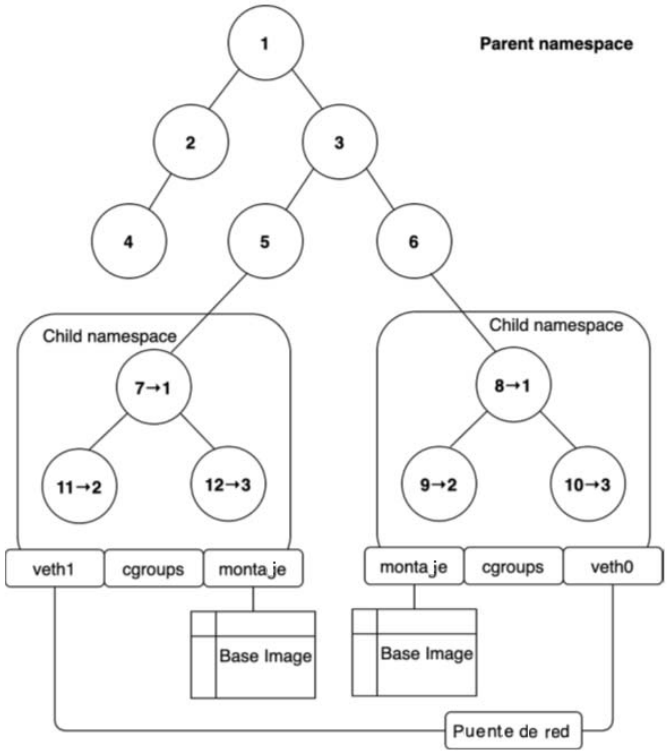

Figura 3: Contenedores: namespaces y cgroups

El primer proceso de un namespace tiene un PID 1 (*Process Identifier 1*). Todos los hijos de este proceso pertenecen automáticamente a este namespace. Todos los procesos de este namespace están aislados de otros namespaces. No pueden ver lo que ocurre fuera de su espacio. Este es el **PID namespace**.

Según este principio, el primer proceso iniciado por el núcleo (init, systemd...) con PID 1 también se encuentra en un namespace: el **parent namespace**. Desde el padre, los procesos dentro de un namespace son vistos como procesos ordinarios, con PIDs "normales". Así, el proceso con PID 1 en un namespace podría tener el PID 12345, visto desde fuera. Por ejemplo, aquí el proceso con PID 1 de este contenedor es el PID 1148 del host en el que se ejecuta:

```
# En el contenedor:
# cat /proc/1/cmdline | tr '\0' ' ' ; echo
/usr/local/openjdk-11/bin/java .

# Vista del servidor:
eni@node2:~$ ps auxww |grep java
root      1148 28.0 21.1 4909116 412356 ?
Ssl  20:22   3:08 /usr/local/openjdk-11/bin/java
```

Los mecanismos de namespace también le permiten modificar el rootfs del proceso, de la misma manera que un chroot. Usted instala todo lo que necesita para que el proceso se ejecute en un directorio y posteriormente, reemplaza el sistema de archivos root / namespace con este directorio. Este es el **mount namespace**.

Cada namespace puede tener una interfaz de red virtual. Esta interfaz suele unida a una interfaz de red del host o con otra interfaz gestionada por un servicio específico. La interfaz virtual recibe su propia dirección IP y garantiza la comunicación entre el namespace y el mundo exterior. Este es el **network namespace**.

Por defecto, los procesos de un namespace comparten usuarios y grupos con los del sistema. Sin embargo, es posible aislarlos, disociarlos o, por el contrario, asociarlos a otros usuarios. De esta forma, es posible tener usuarios con el mismo UID (*User IDentifier*) en dos espacios de nombres, aunque estos usuarios sean diferentes o tener una cuenta root dentro del namespace, que estará asociada a un usuario sin permisoss fuera del namespace. Este es el **user namespace**.

Cada namespace se puede asociar a un cgroup para limitar los recursos de los procesos en el espacio. Este es el **nam cgroup namespace**.

Existen otros dos tipos de espacios de nombres: **IPC** (*InterProcess Communication*), que permite aislar las comunicaciones entre procesos dentro del espacio y **UTS** (*Unix Timesharing System*), que permite cambiar el nombre de host (hostname) y de dominio (*domainname*, NIS (*Network Information System*)) del espacio.

2.4 cgroup

Si coloca procesos en un cgroup, puede controlar los recursos que pueden utilizar. Puede limitar estos procesos a una cantidad determinada de memoria utilizable, por ejemplo 256 MB y a una cantidad determinada de tiempo de procesador, expresada en núcleos o milicores, por ejemplo 500 milinúcleos o 0,5 núcleos. Estos límites son para el espacio de nombres en su conjunto, no límites por proceso.

Los grupos de control se pueden utilizar no sólo para limitar el uso de recursos por parte de un grupo de procesos, sino también para evitar que un grupo de procesos acapare todos los recursos del sistema y, por tanto, canibalice otras aplicaciones y el propio sistema. Así, si un proceso del grupo intenta utilizar más memoria que la cuota asignada, será eliminado por los mecanismos de protección cgroups del núcleo Linux.

2.5 Montaje en unión

Un montaje ne unión (*union mount*) reúne varios sistemas de archivos -normalmente una pila ordenada de directorios- en un único punto de montaje, creando un sistema de archivos virtual a partir de la unión de todos los demás. Cada capa de la pila añade o elimina archivos. Los sistemas de archivos más conocidos son **AUFS** (*Another Union FileSystem*) y **Overlay**. Un sistema de archivos de este tipo también se puede implementar utilizando LVM (*Logical Volume Manager*) o las instantáneas snapshots que ofrecen algunos sistemas de archivos como BTRFS (*B-TRee File System*). En la actualidad, Overlay es el más utilizado.

Este tipo de sistema de archivos ofrece una gran ventaja cuando se monta como sistema de archivos raíz de un contenedor: todas las capas son de sólo lectura, excepto la última capa - un directorio deliberadamente vacío, que es de sólo escritura. Por tanto, el contenedor puede escribir en el punto de montaje sin modificar el sistema de archivos base. Cuando se destruye el contenedor, los cambios se pierden al destruirse la última capa.

Una segunda ventaja de esta tecnología es la posibilidad de aprovechar y compartir determinadas capas. Por ejemplo, si sólo hay una diferencia de 20 MB entre dos imágenes contenedoras de 100 MB, estas dos imágenes sólo ocupan 120 MB, no 200 MB.

2.6 Imagen de la aplicación

Para iniciar una aplicación dentro de un contenedor, todas sus dependencias deben estar instaladas en su sistema de archivos raíz: la aplicación está totalmente aislada y no puede acceder a archivos fuera de su espacio. Un método consiste en copiar en un directorio un árbol estándar de Linux que contenga los archivos básicos (ejecutables, librerías y archivos de configuración), las dependencias de la aplicación que se va a ejecutar en el contenedor y la propia aplicación.

El árbol básico de Linux puede ser el de cualquier distribución.

Puede crear una copia de esta estructura de árbol para crear una imagen base reutilizable. Se puede almacenar, replicar y distribuir. Puede crear catálogos y repositorios. Puede adaptarla a tus necesidades y hacer varias versiones.

Puede montar este árbol como una unión de sólo lectura con un directorio vacío para la escritura y utilizarlo como el sistema de archivos raíz en el espacio de nombres.

Crear un contenedor significa instanciar una imagen del sistema de archivos y, a continuación, iniciar uno o más procesos dentro de un namespace.

2.7 Capas de imágenes

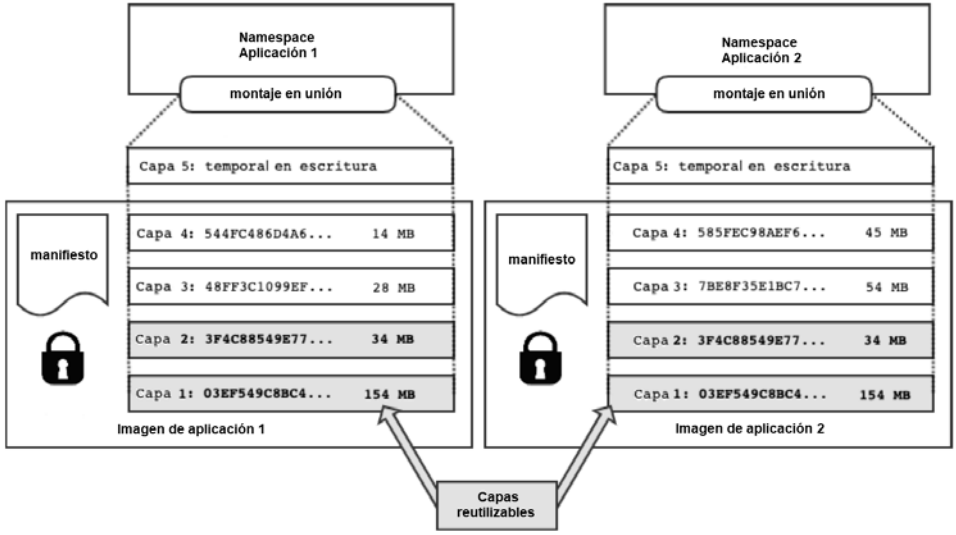

Figura 4: Imágenes en capa y montaje en unión

Cuando construimos una imagen, partimos del mínimo utilizable.

Encima de esta capa mínima, puede añadir actualizaciones de paquetes, luego instalar las dependencias de la aplicación que desea iniciar y, por último, la propia aplicación.

En cada etapa se añade una capa (*layer*) adicional, por ejemplo:

– capa 1: la imagen de base,
– capa 2: actualizaciones,
– capa 3: dependencias de la aplicación,
– capa 4: aplicación.