Sistemas Operativos Avanzados (MUII)
Ejercicio sobre contenedores (5-mayo-2021)
¿De qué están hechos los contenedores?

Planteamiento del trabajo

La virtualización soportada por el sistema operativo, articulada a través del concepto de contenedor, es una de las tecnologías más excitantes de la última década. El objetivo de este trabajo es llegar a conocer cómo el sistema operativo da soporte a esta abstracción basándose, en el caso de Linux, en los mecanismos de grupos de control, espacios de nombres, sistemas de ficheros de tipo unión o de tipo COW y, en general, técnicas de creación de redes mediante software.

El trabajo consiste en escribir un informe sobre cómo usan todas esas tecnologías dos populares soluciones de contenedores: Docker, la solución de contenedores de aplicación por antonomasia, y LXD, una solución de contenedores de sistema.

Docker

Una vez instalado Docker y después de crear en el directorio base de la cuenta un directorio denominado *datos* con varios ficheros, vamos a lanzar el siguiente mandato:

docker run -it --rm -v ~/datos:/data curlimages/curl sh

que descarga y ejecuta la imagen de *curl*, una popular herramienta para la descarga de páginas web, especificando que se ejecute de forma interactiva asociada al terminal *(-it*), que desaparezca cuando se complete su ejecución *(--rm*), que comparta el directorio (volumen, en terminología Docker) *datos* del servidor bajo el nombre de *data*, y que comience ejecutando un *shell* de Bourne. Nótese que la imagen de *curl* está construida sobre una distribución Alpine de Linux, que se caracteriza por su ligereza, pero que no dispone del *bash*. Asimismo, tenga en cuenta que esa imagen está construida de manera que, por defecto, se ejecuta con el UID=100 y el GID=101. Este comportamiento puede modificarse incluyendo la opción *-u* en el mandato *docker run*.

A continuación, vamos a analizar los distintos mecanismos usados para crear ese contenedor.

Ejercicio 1: *cgroups*

Investigue el uso de este mecanismo para implementar el contenedor. Para ello, puede consultar el fichero */proc/PID/cgroup* para conocer con qué grupos de control está vinculado ese proceso y el directorio */sys/fs/cgroup* para identificar qué grupos están asociados al contenedor.

Para poder comprobar de forma aplicada el uso de este mecanismo, se plantean dos acciones:

- Detener la ejecución del contenedor mediante el controlador *freezer*.
- Limitar el número máximo de procesos activos dentro del contenedor usando el controlador *pids*. Nótese que Docker realiza esta acción si se especifica la opción *-pids-limit* en *docker run*.

Escriba un informe que documente el trabajo realizado en este apartado.

Ejercicio 2: *namespaces*

Analice el uso de este mecanismo para dar soporte al contenedor. A tal fin, en primer lugar, compruebe mediante el fichero */proc/PID/ns* qué espacios de nombres comparten el contenedor con el sistema anfitrión ratificándolo para cada uno de los siguientes espacios de nombres usando operaciones que permitan confirmarlo (se resuelve para el primer caso para que sirva de pauta):

- *pid*: el contenedor tiene su propio espacio de nombre de PIDs lo que se puede ratificar ejecutando el mandato *ps* en el contenedor y en el sistema anfitrión. Nótese que los procesos del contenedor son visibles en el anfitrión, aunque con otros PIDs, pero no ocurre así en sentido contrario.
- *uts*.
- *mnt*. Se tratará más adelante el almacenamiento asociado al contenedor; en este punto basta con verificar si se comparte o no ese espacio de nombres.
- *ipc*.
- *net*. Se tratará más adelante la gestión de la red; en este punto basta con verificar si se comparte o no ese espacio de nombres.
- *user*. Se analiza en el siguiente apartado.

Téngase en cuenta que el usuario puede cambiar este comportamiento por defecto especificando en el mandato *docker run* qué espacios de nombres se comparten y cuáles no (por ejemplo*, -pid=host* indica que el contenedor comparte los PIDs con el anfitrión).

Escriba un informe que documente el trabajo realizado en este apartado.

Ejercicio 3: Usuarios

Indique si se comparte el espacio de nombres de usuario. Verifique usando el volumen compartido cómo se ven, en cuanto a quiénes son sus dueños, en el anfitrión los ficheros creados en el contenedor y viceversa. Pruebe con el usuario por defecto de esa imagen y repítalo con el superusuario (opción *-u 0:0*).

Analice este modo de operación por defecto de Docker desde el punto de vista de la seguridad y de la flexibilidad a la hora de compartir datos.

Ejercicio 4: Almacenamiento del contenedor

Como ya conoce, un contenedor recibe dos tipos de sistemas de almacenamiento: el dispositivo de almacenamiento del contenedor propiamente dicho, que contiene la imagen inicial y que se destruye cuando se elimina el contenedor (en ese dispositivo estará montado el directorio raíz del contenedor), y volúmenes, que permiten al contenedor compartir información con el anfitrión y otros contenedores pudiéndose mantener esta información más allá de la vida del contenedor.

Un aspecto clave en la gestión que realiza Docker es que la imagen vinculada con un contenedor está organizada en capas (puede apreciarlo usando el mandato *docker inspect*) lo que permite que varios contenedores puedan compartir algunas de estas capas, optimizando el uso de recursos, de manera que esa compartición se mantiene hasta que un contenedor realiza una modificación obteniendo en ese momento su propia copia del objeto actualizado. Para ello, se usan sistemas de ficheros de tipo unión o con COW.

Investigue qué características tiene el dispositivo de almacenamiento del contenedor (es decir, el dispositivo que incluye el directorio raíz de la jerarquía dentro del contenedor; puede usar *mount* para obtener los dispositivos montados y localizar el asociado al directorio raíz) y descubra cómo se puede escribir en un fichero de ese dispositivo desde el anfitrión.

Ejercicio 5: Conectividad del contenedor

Docker usa los servicios del sistema operativo para crear una red por software de manera que los contenedores activos y el anfitrión se puedan comunicar entre sí.

Describa la información de red vinculada con la conectividad proporcionada por Docker.

Arranque dos contenedores y pruebe a activar un servidor en uno de ellos (*nc -l -p 34567*) y verifique que existe conectividad (*nc IP 34567*) tanto desde el anfitrión como desde el otro contenedor.

LXD

Se trata de una herramienta de gestión de contenedores que, obviamente, no es tan popular como Docker, pero que, además, está ideada para otro campo de aplicación. Mientras que el objetivo de Docker es envolver la ejecución de una aplicación o un servicio en un contexto de ejecución independiente (gestiona contenedores de aplicación), la meta de LXD es crear un ámbito de ejecución separado para la ejecución de una distribución del sistema operativo (gestiona contenedores de sistema), con la que un usuario puede trabajar de forma interactiva, proporcionando a todos los efectos máquinas virtuales ligeras. Nótese que, dado su objetivo diferente, ambas soluciones pueden convivir: se puede usar Docker para ejecutar de forma confinada aplicaciones en una máquina virtual ligera creada por LXD.

Una vez instalado LXD, vamos a lanzar el siguiente mandato:

lxc launch images:alpine/3.13 myalpine

que descarga la imagen de la distribución del sistema operativo especificada y arranca un contenedor denominado *myalpine* asociado a esa imagen.

A continuación, se habilita una carpeta compartida usando este mandato (*compartido* será el nombre que usa internamente LXD para referirse a ese "dispositivo").:

lxc config device add myalpine compartido disk source=~/datos path=/data

Para trabajar con este sistema operativo recién lanzado, se puede usar el mandato *lxc exec*:

lxc exec myalpine sh

Se pide repetir los mismos ejercicios que se plantearon para Docker identificándolos con el sufijo *bis* (*ejercicio 1bis…*). Hay que hacer, sin embargo, dos puntualizaciones.

Ejercicio 3bis: Usuarios

Para preparar el escenario de este ejercicio, arranque un segundo contenedor igual que el primero con el nombre *myalpine2*. A continuación, cree en el primer contenedor, mediante *adduser*, una cuenta para un usuario de nombre *test* con UID 500 (y el GID que le asigne automáticamente).

Indique, en primer lugar, si existe un espacio de nombres de usuario independiente para el contenedor. A continuación, dentro del primer contenedor cree un fichero en el disco compartido siendo *superusuario* y otro como el usuario normal *test*. Analice cómo se visualiza el dueño (UID y GID) de esos ficheros desde el contexto del anfitrión y desde el ámbito del segundo contenedor. A continuación, cree un fichero en el contexto del anfitrión siendo su usuario normal y otro en el rol de *superusuario* y compruebe cómo se visualiza en el ámbito de los contenedores. Para este análisis, tenga cuenta el valor de los ficheros */proc/PID/uid_map* y */proc/PID/gid_map*.

Ejercicio 4bis: Almacenamiento del contenedor

Con respecto al almacenamiento, cuando se instala LXD, se crea una zona de almacenamiento por defecto (*default storage pool*) para todos los contenedores que puede corresponder a un disco completo, a una partición o, por defecto, a un fichero usado como *loop device*. Cuando se crea un nuevo contenedor, se habilita un volumen lógico, ya sea usando un gestor de volúmenes, como *LVM*, o sistemas de ficheros que incluyan la funcionalidad de gestionar volúmenes, tales como *btrfs* o *zfs*, dentro de esa zona de almacenamiento, que estará destinado a ser el disco del nuevo contenedor.
Este ejercicio simplemente plantea obtener información sobre el dispositivo raíz del contenedor.

Plazo y modo de entrega

El plazo de entrega del trabajo es el 15 de junio de 2021.

La entrega se realiza en *triqui* ejecutando el mandato:

entrega.soa contenedores.2021

Este mandato realizará la recolección del directorio *~/DATSI/SOA/contenedores.2021* de los ficheros *autor.txt*, con los datos del alumno, y *memoria.pdf*, que debe incluir la solución del ejercicio.