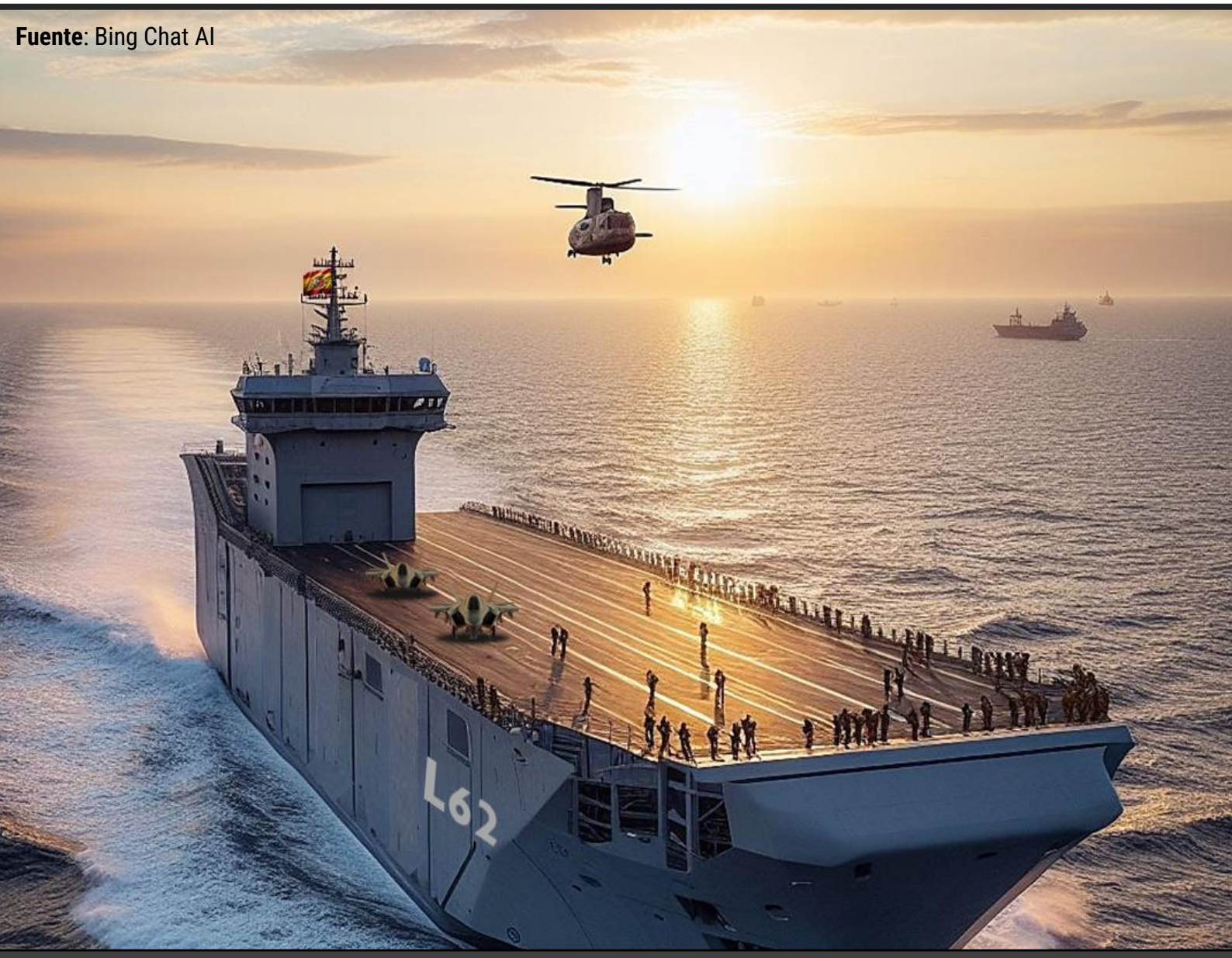
Fuente: Bing Chat AI



Bloque 3.1. Contenedores de Sistemas Operativos

LXD

MÁSTER EN INGENIERÍA WEB 2023 – 2024

Administración de Sistemas Operativos v1.2 *("L-62 Princesa de Asturias")*

© José Manuel Redondo López. Universidad de Oviedo

Contenido

Nivel 1 (Cabo): Instalación y uso básico 6
¿Qué necesito para hacer este bloque de actividades? 6
N1LXD_INSTALAR: Instalación del software *LXD* *(0,2 puntos)* 6
Descripción de la actividad 6
Resultados esperados 6
Información necesaria para su realización 6
N1LXD_RUNCONT: Arranque de un contenedor preconstruido *(0,1 puntos)* 10
Descripción de la actividad 10
Resultados esperados 10
Información necesaria para su realización 10
N1LXD_INTERACT: Conexión e interacción con un contenedor *(0,1 puntos)* 11
Descripción de la actividad 11
Resultados esperados 11
Información necesaria para su realización 11
N1LXD_WEBCONT: Crear un contenedor con un servidor web *(0,1 puntos)* 11
Descripción de la actividad 11
Resultados esperados 12
Información necesaria para su realización 12
N1LXD_STARTSTOP: Parada de un contenedor y limpieza de contenedores *(0,1 puntos)* 12
Descripción de la actividad 12
Resultados esperados 12
Información necesaria para su realización 12
Nivel 2 (Sargento): Resolución de problemas sencillos frecuentes 13
N2LXD_GESTINST: Gestión de instancias *(0,1 puntos)* 13
Descripción de la actividad 13
Resultados esperados 13
Información necesaria para su realización 13
N2LXD_CUSTOMCONT: Creación de un contenedor personalizado *(0,2 puntos)* 14
Descripción de la actividad 14
Resultados esperados 14
Información necesaria para su realización 14
N2LXD_SSH: Conexión vía SSH a un contenedor *(0,1 puntos)* 14

Descripción de la actividad 14
Resultados esperados 15
Información necesaria para su realización 15
N2LXD_INFRAWEB: Infraestructura para web y base de datos con contenedores *(0,3 puntos)* **15**
Descripción de la actividad 15
Resultados esperados 15
Información necesaria para su realización 15
Nivel 3 (Teniente de navío): Resolución de situaciones más avanzadas 16
N3LXD_INITCONF: Aplicando configuraciones en el arranque de una instancia *(0,1 puntos)* **16**
Descripción de la actividad 16
Resultados esperados 16
Información necesaria para su realización 16
N3LXD_FICHEROSEXT: Copia de ficheros externos a un contenedor *(0,2 puntos)* **16**
Descripción de la actividad 16
Resultados esperados 17
Información necesaria para su realización 17
N3LXD_CREARVOL: Creación de volúmenes de contenedor *(0,2 puntos)* **17**
Descripción de la actividad 17
Resultados esperados 17
Información necesaria para su realización 17
N3LXD_PROVISION: Provisionamiento de contenedores *(0,2 puntos)* **18**
Descripción de la actividad 18
Resultados esperados 18
Información necesaria para su realización 18
N3LXD_LIMITES: Actualización de la configuración/límites de una instancia en ejecución *(0,2 puntos)* **18**
Descripción de la actividad 18
Resultados esperados 18
Información necesaria para su realización 19
Nivel 4 (Capitán de navío): Explotación de contenedores de SO 20
N4LXD_PROFILE: Configuración de contenedores a partir de código: Manejo de perfiles *(0,3 puntos)* **20**
Descripción de la actividad 20
Resultados esperados 20
Información necesaria para su realización 20

N4LXD_EXPORT: Exportando servicios de contenedores a clientes externos con perfiles *(0,3 puntos)* **22**
Descripción de la actividad 22
Resultados esperados 22
Información necesaria para su realización 23
N4LXD_LANCONF: Configuración de redes locales entre contenedores *(0,3 puntos)* **23**
Descripción de la actividad 23
Resultados esperados 23
Información necesaria para su realización 24
N4LXD_LXDGUI: Aplicaciones con GUIs *(0,2 puntos)* **25**
Descripción de la actividad 25
Resultados esperados 25
Información necesaria para su realización 25
N4LXD_INFRAWEB: Infraestructura para web y base de datos con contenedores *(0,3 puntos)* **26**
Descripción de la actividad 26
Resultados esperados 27
Información necesaria para su realización 27
N4LXD_PLATMIW: Instalación de plataformas especializadas de otras asignaturas *(0,5 puntos)* **27**
Descripción de la actividad 27
Resultados esperados 27
Información necesaria para su realización 28
Nivel 5 (Almirante): Operaciones avanzadas con contenedores de SO **29**
N5LXD_CLOUDINIT: Escribir un perfil *cloud-init* *(0,2 puntos)* **29**
Descripción de la actividad 29
Resultados esperados 29
Información necesaria para su realización 29
N5LXD_CUSTOMLXD: Creación de una imagen personalizada de LXD *(0,4 puntos)* **30**
Descripción de la actividad 30
Resultados esperados 30
Información necesaria para su realización 30
N5LXD_DOCKERLXD: *Docker y LXD* *(0,2 puntos)* **30**
Descripción de la actividad 30
Resultados esperados 30
Información necesaria para su realización 30

Insignias y Autoevaluación 32

Nivel 1 (Cabo): Instalación y uso básico

¿Qué necesito para hacer este bloque de actividades?

Para realizar este bloque de actividades necesitas una máquina física o virtual *Ubuntu Linux 18.04* o superior, **como por ejemplo las que se puedan haber creado con el módulo de *Vagrant***. Necesitas:

- **Un software de virtualización instalado**: en principio sirve cualquiera. Se recomienda *VirtualBox*, como en el caso de *Vagrant*.
- **Núcleos de CPU**: Con 2 se puede trabajar perfectamente, un extra siempre es bienvenido
- **RAM**: Mínimo 2, mejor 4 o más
- **SO**: Un *Ubuntu 18.04* o superior. Las pruebas se han hecho con la versión *18.04*.

N1LXD_INSTALAR: Instalación del software *LXD* *(0,2 puntos)*

Descripción de la actividad

Consiste en **hacer una instalación funcional de LXD**.

Resultados esperados

Eres capaz de hacer una instalación operativa de LXD con un *backend* de almacenamiento ZFS

Información necesaria para su realización

Preparación de la instalación

Esta actividad consiste en instalar una versión funcional de LXD. No obstante, **previamente vamos a instalar el sistema de ficheros ZFS**, que es una de las opciones más adecuadas que tenemos para gestionar el almacenamiento de los contenedores. Para ello necesitamos crear en la máquina virtual de pruebas **un disco duro nuevo dinámico de 120Gb** que se usará como una partición independiente para guardar todos los datos de los contenedores de forma separada a los datos del *host*.

ZFS es un sistema de ficheros que tiene muchas ventajas que lo hacen muy adecuado para desplegar infraestructuras de contenedores de SO grandes y, en general, **guardar grandes cantidades de datos**:

- Tiene una gran capacidad de almacenamiento, hasta 16 exabytes.
- Integra el sistema de ficheros y el administrador de volúmenes.
- Usa una estructura basada en bloques para mejorar la integridad y el rendimiento en el acceso a los datos.
- Permite crear sistemas de ficheros “ligeros” dentro de un espacio de almacenamiento virtual.

- Ofrece funciones como compresión, cifrado, deduplicación y copia sobre escritura.

ZFS es un sistema de ficheros muy avanzado y potente que se puede usar en diferentes sistemas operativos y entornos. Aunque en nuestro escenario es excesivo, conviene usarlo desde el principio para tener una idea de **cómo se usa LXD en un entorno de producción real**.

Aunque esta asignatura no es de SSOO, sí que tenemos que conocer un poco ZFS para poder usarlo de manera adecuada. Eso implica conocer el concepto de ***pool* de ZFS**. Un *pool* ZFS es una **colección de uno o más dispositivos virtuales** que sirven como contenedor de datos de alto nivel en un sistema ZFS. Se puede usar para crear sistemas de archivos o volúmenes que comparten el espacio disponible en el pool. Algunas de las ventajas de usar un pool ZFS son:

- Permite una **gestión sencilla y flexible** del almacenamiento, ya que se puede añadir o quitar dispositivos sin necesidad de particionar o formatear.
- Ofrece una **alta integridad y seguridad de los datos**, ya que usa técnicas como la copia sobre escritura, la verificación de la suma de comprobación y la corrección de errores.
- Mejora el **rendimiento y la eficiencia del sistema**, ya que usa la memoria RAM como caché y permite funciones como la compresión, la deduplicación y el cifrado.

Aunque no es necesario para esta asignatura, puede que si lo sea para tu futuro profesional. Si quieres saber más sobre el sistema de archivos ZFS, puedes consultar las siguientes páginas:

- **ZFS: Qué es, para qué sirve y cómo usarlo**, que te explica las características y ventajas de ZFS frente a otros sistemas de archivos tradicionales. https://www.profesionalreview.com/2022/07/26/zfs/
- **ZFS sistema de archivos para servidores - Instalación y Configuración**, que te muestra cómo instalar y configurar ZFS en Linux y cómo crear y gestionar pools y sistemas de archivos. https://www.redeszone.net/tutoriales/servidores/sistema-archivos-zfs-servidores/
- **Actividad: ZFS**, que te propone una serie de ejercicios prácticos para aprender a usar ZFS en un entorno virtual. https://elpuig.xeill.net/Members/vcarceler/asix-m06/a6

Para empezar a instalar LXD de la forma vista, **tomamos nota del disco a usar para el *pool* ZFS**, teniendo mucho cuidado de no confundirse de disco. Para eso ejecutamos: `ls -la /dev/disk/by-id/`

```
ssiuser@vagrant:~$ ls -la /dev/disk/by-id
total 0
drwxr-xr-x 2 root root 200 Jul 16 20:08 .
drwxr-xr-x 6 root root 120 Jul 16 20:08 ..
lrwxrwxrwx 1 root root   9 Jul 16 20:08 ata-VBOX_HARDDISK_VB8e76489b-91664c56 -> ../../sda
lrwxrwxrwx 1 root root  10 Jul 16 20:08 ata-VBOX_HARDDISK_VB8e76489b-91664c56-part1 -> ../../sda1
lrwxrwxrwx 1 root root   9 Jul 16 20:08 ata-VBOX_HARDDISK_VBd278748e-1d8f2018 -> ../../sdb
lrwxrwxrwx 1 root root  10 Jul 16 20:08 dm-name-vagrant--vg-root -> ../../dm-0
lrwxrwxrwx 1 root root  10 Jul 16 20:08 dm-name-vagrant--vg-swap_1 -> ../../dm-1
lrwxrwxrwx 1 root root  10 Jul 16 20:08 dm-uuid-LVM-6Q0MIBcjFfM7QJc1QS8u2luDtqkQU2A5QWqJLursOLQFjTAq
JpOqqWFq6gQYRkA2 -> ../../dm-0
lrwxrwxrwx 1 root root  10 Jul 16 20:08 dm-uuid-LVM-6Q0MIBcjFfM7QJc1QS8u2luDtqkQU2A5sxU2QIV3ovdADRTf
1c3vde7vCkTRuPjm -> ../../dm-1
lrwxrwxrwx 1 root root  10 Jul 16 20:08 lvm-pv-uuid-t2mdkA-YYUs-YGO6-LDxu-I8rk-DdPc-dWScXR -> ../../
sda1
```

El disco parece ser el llamado `sdb`. Como doble verificación, **comprobamos que es el disco que hemos creado para este fin** con `sudo fdisk -l`, mirando su tamaño, como aparece en la siguiente imagen. Aquí confirmamos que se trata efectivamente del `sdb` y por tanto su *path* completo en la ruta `by-id` (el que tendremos que usar en la inicialización de ZFS) será: `/dev/disk/by-id/ata-VBOX_HARDDISK_VBd278748e-1d8f2018` (en cada MV será uno distinto, puesto que **el ID se genera aleatoriamente**).

```
ssiuser@vagrant:~$ sudo fdisk -l
Disk /dev/sda: 100 GiB, 107374182400 bytes, 209715200 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x4ec46e60

Device     Boot Start       End   Sectors Size Id Type
/dev/sda1  *     2048 134215679 134213632  64G 8e Linux LVM


Disk /dev/sdb: 120 GiB, 128849018880 bytes, 251658240 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes


Disk /dev/mapper/vagrant--vg-root: 63 GiB, 67687677952 bytes, 132202496 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes


Disk /dev/mapper/vagrant--vg-swap_1: 980 MiB, 1027604480 bytes, 2007040 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

Una vez tomemos nota de esto, ya podemos instalar las dependencias necesarias para usar ZFS en nuestro servidor *Ubuntu*: `sudo apt install zfsutils zfsutils-linux`. LXD está disponible para varios sistemas operativos *Linux* y para *Windows*, y podemos encontrar distintas formas de instalación documentadas en su web oficial. En nuestras pruebas vamos a usar una máquina virtual *Ubuntu*. En *Ubuntu 18.04* el paquete está en el gestor `apt` (`sudo apt install lxd`), mientras que en versiones posteriores se trata de un paquete `snap`: `sudo snap install lxd`. Hecho esto, agregamos el usuario que usemos habitualmente al grupo `lxd` para poder manejar los contenedores sin `sudo`: `sudo usermod --append --groups lxd <tu usuario>`. Terminamos **cerrando nuestra sesión y volviendo a logearnos con nuestro usuario**.

CUIDADO: Cualquier miembro del grupo `lxd` tiene acceso al *daemon* de LXD y puede suponer un riesgo de seguridad, por lo que no se puede asignar la pertenencia a este grupo a la ligera. Para saber más de seguridad LXD se recomienda consultar: https://linuxcontainers.org/lxd/docs/master/security

Configurando Ubuntu LXD

Tras la instalación de LXD necesitamos hacer una configuración inicial del mismo contestando a una serie de preguntas que nos harán tras ejecutar `lxd init`.(o `sudo lxd init` si no hemos añadido nuestro usuario al grupo `sudoer`, en cuyo caso deberemos anteponer `sudo` a la mayoría de los comandos de `lxd`). Podemos dar *intro* a todas las preguntas para dejar la configuración por defecto, pero es necesario darle la ruta del dispositivo a usar con ZFS que vimos antes

```
Would you like to use LXD clustering? (yes/no) [default=no]:
Do you want to configure a new storage pool? (yes/no) [default=yes]:
Name of the new storage pool [default=default]: zfs
Name of the storage backend to use (btrfs, dir, lvm, zfs) [default=zfs]:
Create a new ZFS pool? (yes/no) [default=yes]:
Would you like to use an existing block device? (yes/no) [default=no]: yes
Path to the existing block device: /dev/disk/by-id/<el disco que has localizado antes>
Would you like to connect to a MAAS server? (yes/no) [default=no]:
```

```
Would you like to create a new local network bridge? (yes/no) [default=yes]:
What should the new bridge be called? [default=lxdbr0]:
What IPv4 address should be used? (CIDR subnet notation, “auto” or “none”) [default=auto]:
What IPv6 address should be used? (CIDR subnet notation, “auto” or “none”) [default=auto]:
Would you like LXD to be available over the network? (yes/no) [default=no]:
Would you like stale cached images to be updated automatically? (yes/no) [default=yes]
Would you like a YAML "lxd init" preseed to be printed? (yes/no) [default=no]:
```

El significado de estas opciones aparece en la tabla siguiente. Como veremos luego, **LXD soporta tanto contenedores como máquinas virtuales**. **Cuando se menciona la palabra “instancias” en este boletín se refiere a ambos**. Si no queremos configurar nada podemos optar por hacer: `sudo lxd init -minimal`, pero perderemos funcionalidades, por lo que no se recomienda.

Característica	Descripción	Opciones de configuración	Para saber más…
Clustering	Un clúster puede combinar varios servidores LXD. Usan la misma base de datos distribuida y se pueden gestionar de forma uniforme usando el cliente de LXD (`lxc`) o su API REST. Esto no lo usaremos en este curso por no hacer uso excesivo de recursos	`default=no;` Si se pone a `yes`, podrás conectarte a un clúster existente o crear uno nuevo	Documentación de LXD: Clustering
MAAS server	MAAS es un software *open source* que te permite crear un *data center* en tus servidores	`default=no;` Si se pone a `yes`, podrás conectarte a un servidor MAAS existente especificando nombre, URL y una *API key*	- maas.io - MAAS - How to manage VM hosts
Network bridge	Proporciona acceso a la red a las instancias de contenedores LXD	Puedes especificar un *bridge* o interface existente, o dejar que LXD cree uno nuevo (**recomendado**). Se pueden crear nuevos *bridges* y asignarlos a instancias luego	Documentación de LXD: - Networks - Network interface
Storage pools	Las instancias, etc. se guardan en estas *pools*, por lo que es necesario crear una	Para producción se recomienda usar una partición o disco vacío dedicado a LXD, a ser posible ZFS (como en este curso) o *btrfs*. Se pueden crear pools adicionales luego	Documentación de LXD: - Storage - Comparison of methods - Backend comparison chart
Network access	Permite acceder al servidor LXD a través de la red (**CUIDADO**: Esto no tiene nada que ver con las instancias, ¡es el servidor de estas!). **No lo usaremos en este curso**	`default=no;` Si se pone a `yes`, podrás conectarte a un servidor a través de la red. Puedes establecer una *password* o aceptar	

		manualmente certificados de cliente.	
Automatic image update	Las imágenes que se descarguen de servidores se actualizarán automáticamente si hay nuevas versiones	`default=yes;` Esto significa que LXD actualizará las imágenes regularmente si al servidor se suben versiones nuevas	Documentación de LXD: Image handling
YAML lxd init preseed	Muestra un resumen de las opciones de configuración en el terminal	`default=no`	

Hecho todo esto, simplemente es necesario usar el comando `lxd` para comprobar que está disponible sin errores.

N1LXD_RUNCONT: Arranque de un contenedor preconstruido *(0,1 puntos)*

Descripción de la actividad

Conocer las herramientas básicas de **listado y arranque de instancias** de LXD.

Resultados esperados

Eres capaz de listar las instancias disponibles en los servidores de imágenes por defecto y lanzar instancias de distintos servidores

Información necesaria para su realización

Listar todas las imágenes disponibles

Para listar imágenes disponibles para descarga de los repositorios estándar vistos en la teoría ejecutamos el siguiente comando, usando `less` **debido a la gran cantidad de ellas existente**: `lxc image list images: | less`. Debido a ello, lo más aconsejable es filtrar con `grep` y un *pipe* de *Linux* para localizar las de un tipo determinado. Por ejemplo : `lxc image list images: | grep Ubuntu`

Lanzando instancias

El comando `lxc launch` permite lanzar instancias si especificamos la imagen que queremos lanzar y el nombre que le queremos dar. El nombre de las imágenes es uno de los que obtenemos de la lista anterior. **Recuerda**: Si no tenemos un servidor de imágenes, se usará uno de los que viene por defecto mencionados en la teoría.

- **Para lanzar un contenedor**: `lxc launch <image_server>:<image_name> <instance_name>`. Ejemplo: `lxc launch images:ubuntu/20.04 ubuntu-container`

Para lanzar una máquina virtual (necesitamos instalar `qemu`, **y no lo usaremos en esta asignatura**): `lxc launch <image_server>:<image_name> <instance_name> --vm`. Ejemplo: `lxc launch images:ubuntu/20.04 ubuntu-vm --vm`

Para completar esta actividad debes lanzar tres instancias: una *Ubuntu 18* o superior, una *Alpine Linux* de la última versión disponible y **añadir un servidor de imágenes** como se indica aquí: https://cloud-images.ubuntu.com/minimal/releases/bionic/ para lanzar una instancia de `ubuntu-minimal`

N1LXD_INTERACT: Conexión e interacción con un contenedor *(0,1 puntos)*

Descripción de la actividad

Consiste en aprender a **usar un contenedor LXD de forma interactiva**.

Resultados esperados

Eres capaz de ejecutar comandos en un contenedor o entrar en el mismo con una sesión de usuario interactiva

Información necesaria para su realización

Se pueden ejecutar comandos en un contenedor de dos formas:

- Pasándolos directamente desde el host: `lxc exec <instance_name> -- <command>`. Ejemplo: `lxc exec ubuntu-container -- apt-get update`
- Accediendo a un *shell* y tratándolo como un sistema *Linux* cualquiera: `lxc exec <instance_name> -- /bin/bash`. Por defecto se hace *login* como `root`, aunque se puede cambiar de usuario así: `lxc exec <instance_name> -- su --login <user_name>`, **si bien es posible que muchos contenedores necesiten crear usuarios primero para poder hacerlo**.
    - Otra opción para entrar en sesión es: `lxc console <instance_name>` y **dar INTRO después si no tenemos salida por pantalla**. Para salir de sesión hay que pulsar `Ctrl+a-q`.
    - **El problema con esta segunda opción** es que a priori no sabremos de una combinación de usuario y *password* válida para estos contenedores, así que hay que usar la primera opción con `exec` para asignarle una conocida a los usuarios existentes o crear usuarios nuevos. Muchos contenedores vienen con un usuario creado por defecto en función de su distribución. Por ejemplo, los contenedores *Ubuntu* vienen con el usuario `ubuntu` ya creado.

N1LXD_WEBCONT: Crear un contenedor con un servidor web *(0,1 puntos)*

Descripción de la actividad

Consiste en **crear un servidor web funcional dentro de un contenedor LXD**.

Resultados esperados

Eres capaz de arrancar un servidor web con una web por defecto en un contenedor *Linux* LXD y acceder al mismo desde el *host*

Información necesaria para su realización

Para comprobar que comprendes adecuadamente como trabajar con un contenedor esta actividad te pide **arrancar un contenedor *Linux* de los disponibles** (se recomienda uno *Ubuntu*, pero puede ser el que quieras), instalar en el mismo el servidor web `apache2` (o *Nginx*, como tú quieras) y acceder al mismo con la IP que se le asigna al contenedor automáticamente desde el *host*.

NOTA: Recuerda que para saber la IP de un contenedor puedes usar el comando `ifconfig` que está disponible si instalas en el contenedor el paquete `net-tools`.

N1LXD_STARTSTOP: Parada de un contenedor y limpieza de contenedores *(0,1 puntos)*

Descripción de la actividad

Consiste en aprender a **arrancar y parar un contenedor LXD** a voluntad.

Resultados esperados

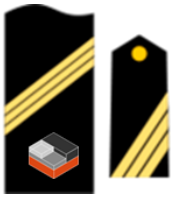
Eres capaz de hacer operaciones de gestión básicas con contenedores LXD

Información necesaria para su realización

En esta actividad se trata de practicar el arranque, parada y borrado de contenedores LXD:

- **Arranque**: `lxc start <instance_name>`. Si la instancia no existe o está en ejecución dará un error
- **Parada**: `lxc stop <instance_name>`. De igual forma dará un error si la instancia no está en ejecución o no existe
- **Reinicio**: Algunos cambios en los contenedores requieren un reinicio de estos, que se puede conseguir fácilmente con `lxc restart <instance_name>`

Si no necesitamos más una instancia, es buena idea **borrarla**, debido al espacio que pueden llegar a ocupar. Es necesario pararla antes de borrarla: `lxc delete <instance_name>`. Esto **es una operación peligrosa porque se borra todo el contenido de forma permanente**. Para evitar accidentes se pueden seguir estas instrucciones: https://linuxcontainers.org/lxd/advanced-guide/#prevent-accidental-deletion-of-an-instance



Nivel 2 (Sargento): Resolución de problemas sencillos frecuentes

N2LXD_GESTINST: Gestión de instancias *(0,1 puntos)*

Descripción de la actividad

Consiste en aprender **operaciones de gestión básicas** sobre instancias.

Resultados esperados

Eres capaz de ver todas las instancias activas y filtrarlas por criterios diferentes para controlarlas.

Información necesaria para su realización

En esta actividad vamos a ver una serie de opciones de gestión útiles y típicas de `lxc`, la herramienta en línea de comandos de LXD. Para una lista completa de comandos basta con escribir el nombre, mientras que para obtener ayuda y detalles sobre un comando concreto sirve con `lxc <comando> --help`

Listado y filtrado

Estos comandos permiten controlar todas las instancias activas en un *host*:

- **Listar todas las instancias de contenedores**: `lxc list`
- **Listar instancias filtradas por algún criterio**. Ejemplos.:
  - Búsqueda de imágenes: Admite nombres para filtrarlas mejor; Ej.: `lxc image list images: debian`, `lxc image list images: debian amd64`
  - Estado actual (parada, en ejecución...): `sudo lxc list volatile.last_state.power=RUNNING` (o `STOPPED`)
  - Por tipo de SO: `sudo lxc list image.os=Ubuntu`
  - Por nombre (admite expresiones regulares): `lxc list ubuntu.*`

Información de instancias individuales

Para obtener información acerca de una instancia debemos hacer `lxc info <instance_name>`. La **lista de atributos** de un contenedor con los que podemos hacer las búsquedas filtradas anteriores la podemos obtener a partir de la información que da este comando.

Por otro lado, si la instancia ha tenido algún problema podemos diagnosticarlo mirando las entradas de su *log*. Para hacerlo ejecutamos `lxc info <instance_name> --show-log`

N2LXD_CUSTOMCONT: Creación de un contenedor personalizado

(0,2 puntos)

Descripción de la actividad

Consiste en **crear un contendor LXD con una serie de *software* a tu gusto**, que se adapte a tus necesidades

Resultados esperados

Eres capaz de crear un contenedor a partir de una imagen existente que tenga el *software* y usuarios que necesites para aplicaciones concretas y usarlo como base para futuros contenedores, creando una imagen a partir de él.

Información necesaria para su realización

En esta actividad se trata de arrancar un contenedor de una imagen existente (se recomienda *Ubuntu*) y **hacerle una instalación básica de algunas herramientas típicas**

- Dejar el *Ubuntu* actualizado (`apt update`, `apt full-upgrade`)
- Instalar *Apache 2* (`apt install apache2`)
- El paquete `net-tools` (para `ifconfig`)
- Los paquetes `nano`, `curl` y `nmap` (para usar las órdenes correspondientes)
- Instalar un servidor `ssh` para poder conectarse a él por esa vía
- Crear un usuario que se llame como tu UO y el paquete `sudo`. Hacer posteriormente que dicho usuario sea *sudoer*

Hecho esto, la actividad requiere el uso de `lxc copy` y `lxc publish` para publicar una copia de un contenedor como imagen **en nuestro repositorio local** (nunca en el repositorio público) que podamos usar para crear nuevos contenedores.

Conviene destacar que, al copiar un contenedor, el resultado es otro en estado `STOPPED`, que debemos arrancar primero antes de poder usarlo. Por otro lado, antes de hacer un `publish` **es necesario parar el contenedor**, salvo que usemos la opción `--force` que lo para y reinicia en el proceso. En el proceso de publicación también es muy conveniente usar la opción `--alias` para darle un nombre al contenedor publicado que sea reconocible para usarlo después fácilmente. Se pueden ver los contenedores publicados localmente con `lxc image list`.

N2LXD_SSH: Conexión vía SSH a un contenedor *(0,1 puntos)*

Descripción de la actividad

Consiste en **establecer una conexión interactiva segura** con un contenedor LXD.

Resultados esperados

Eres capaz de comprobar que un contenedor de SO con el *software* adecuado admite conexiones vía SSH

Información necesaria para su realización

En el contenedor anterior hemos instalado un servidor `ssh`, por lo que al convertirlo a imagen y luego crear contenedores a partir de ella, lo esperable es que podamos conectarnos al mismo vía `ssh`. Aunque LXD proporciona una forma propia de tener una terminal en un contenedor, conectarse vía `ssh` es necesario para otros servicios (`scp`, el *Ansible* del bloque 4…), por lo que **no es raro tener que instalarlo**.

En esta actividad debemos comprobar que este servidor `ssh` instalado está activo y que podemos usarlo para conectarnos al contenedor anterior, lo cual nos abre la opción a otras posibilidades futuras. Recuerda la sintaxis del contenedor: `ssh <nombre de un usuario del contendor>@<IP asignada automáticamente al crear el contenedor>`

N2LXD_INFRAWEB: Infraestructura para web y base de datos con contenedores *(0,3 puntos)*

Descripción de la actividad

Consiste en arrancar contenedores LXD con ***software* típico presente en servicios que se ofrezcan vía web**.

Resultados esperados

Eres capaz de arrancar varios contenedores distintos y configurarlos. Puedes responder a esta pregunta: *¿Permanecen estos contenedores arrancados cuando la máquina que los contiene se reinicia?*

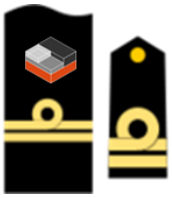
Información necesaria para su realización

Ahora que sabemos arrancar contenedores e interaccionar con ellos, en esta actividad se pretende afianzar esto arrancando los siguientes sobre un mismo *host*:

- Contenedor oficial `ubuntu` con `mysql` instalado
- Contenedor oficial `ubuntu` con `apache2` instalado
- Contenedor oficial `debian` en el que se instale un servidor `ssh`

Hecho esto, reinicia la máquina virtual y **comprueba si estos tres contenedores están operativos de nuevo** una vez entres en sesión de nuevo en la máquina virtual.

NOTA: Si usáis un contenedor con un *Linux* tipo *Red Hat* (*Oracle Linux*, *RHEL*, *CentOS*...) es posible **que, al arrancar el contenedor, LXD no le de una IP de acceso**. Esto es un problema conocido y existe una solución aquí mientras no se arregla de forma "oficial": https://discuss.linuxcontainers.org/t/centos8-containers-unable-to-automatically-get-ipv4-addresses-after-update/11273?page=2



Nivel 3 (Teniente de navío): Resolución de situaciones más avanzadas

N3LXD_INITCONF: Aplicando configuraciones en el arranque de una instancia *(0,1 puntos)*

Descripción de la actividad

Consiste en aplicar **configuraciones iniciales a los contenedores** LXD arrancados.

Resultados esperados

Eres capaz de aplicar opciones de configuración a instancias en el arranque, bien a través de comandos o de un fichero YAML que se le pase

Información necesaria para su realización

Esta actividad requiere practicar con las **distintas formas de pasar opciones de configuración a una instancia**. Para ello es necesario probar cada una de las siguientes:

- Se pueden especificar opciones de configuración concretas al lanzar una instancia usando la opción `--config` o `-c` en un `lxc launch`. Esto solo admite opciones de instancia, no dispositivos. Por ejemplo, para **limitar una instancia** a 2 CPU y 256Mb de RAM hacemos: `lxc launch images:ubuntu/20.04 ubuntu-limited -c limits.cpu=2 -c limits.memory=256MiB`
- Si no contamos con una configuración en `.yml` ya hecha, podemos extraer la de una instancia existente con `lxc config show <instance_name>`. Esto muestra una configuración en `.yml` que podemos convertir fácilmente a un archivo que luego podemos usar para crear nuevos contenedores con la configuración que queramos

NOTA: Posteriormente veremos el uso de perfiles (*Profiles*) para mejorar este aspecto. Por el momento, para esta actividad nos sirve saber a qué podemos aplicarle límites con las opciones `-c`

N3LXD_FICHEROSEXT: Copia de ficheros externos a un contenedor *(0,2 puntos)*

Descripción de la actividad

Consiste en aprender a **meter ficheros** en un contenedor desde un host o a **extraerlos** del contenedor hacia el host.

Resultados esperados

Eres capaz de copiar o extraer ficheros o directorios de un contenedor

Información necesaria para su realización

El tráfico bidireccional de ficheros host<->contenedor es posible en todo momento, y en esta actividad se trata de probar cómo hacerlo en ambas direcciones con el fichero o directorio que desees:

Extracción de ficheros:

- Para extraer archivos de un contenedor se puede hacer lo siguiente: `lxc file pull <nombre_instancia>/<path_to_file> <location_on_host>`.
  - Por ejemplo, para extraer el fichero `/etc/hosts` al directorio actual del *host*: `lxc file pull ubuntu-container/etc/hosts`.
- También se puede hacer un *pipe* y concatenar la salida a otro comando, por ejemplo: `lxc file pull ubuntu-container/var/log/syslog - | less`
- Si se quiere extraer un directorio y todos sus contenidos: `lxc file pull -r <instance_name>/<path_to_folder> <location_on_host>`. Una alternativa a `-r` es `--recursive`.

Copia de ficheros:

- Para copiar archivos a un contenedor: `lxc file push <location_on_host> <instance_name>/<path_to_file>`
- Si queremos copiar un directorio con todos sus contenidos: `lxc file push -r <location_on_host> <instance_name>/<path_to_folder>`. Una alternativa a `-r` es `--recursive`.

N3LXD_CREARVOL: Creación de volúmenes de contenedor *(0,2 puntos)*

Descripción de la actividad

Consiste en **crear y usar volúmenes de almacenamiento** que se puedan adjuntar a contenedores LXD.

Resultados esperados

Eres capaz de crear un volumen y adjuntarlo a una instancia en una ruta conocida de la misma. Puedes contestar a las siguientes preguntas:

- *¿Para qué crees que te puede servir la capacidad de añadir o quitar volúmenes "en caliente" a las instancias de LXD?*
- *¿Se te ocurre un caso de uso vinculado a la web donde puedas usar un volumen que tenga un contenido ya creado con distintos contenedores?*

Información necesaria para su realización

Usando los contenidos de teoría, esta actividad requiere **crear un nuevo volumen de almacenamiento** de tipo `filesystem` y adjuntarlo a una instancia de contenedor en ejecución. Para ello, ten en cuenta que

necesitas primero ejecutar `lxc storage list` para ver el nombre de la *pool* donde se guardarán los nuevos volúmenes (`zfs` si has puesto el mismo nombre que al principio de este documento). Los volúmenes actuales y los nuevos que crees deberían aparecer al ejecutar `lxc storage volume list zfs` (o el nombre de la *pool* que hayas usado).

Una vez hecho esto, podemos usar `lxc storage volume create <nombre de la pool (Ej. zfs)> <nombre único para el volumen>` para **crearlo**. Para **añadirlo a una instancia en marcha** podemos usar `lxc storage volume attach <nombre de la pool> <nombre del volumen> <nombre de la instancia> <ruta dentro del sistema de ficheros de la instancia donde añadir el volumen>`. Ej.: `lxc storage volume attach zfs mivolumen ubuntu-custom /back`

N3LXD_PROVISION: Provisionamiento de contenedores *(0,2 puntos)*

Descripción de la actividad

Consiste en **instalar *software* a voluntad de forma automatizada** en un contenedor LXD.

Resultados esperados

Eres capaz de provisionar un contenedor automáticamente con un *shell script* que previamente copies en él. Puedes contestar a esta pregunta: *¿Entiendes ahora como puedes crear fácilmente instancias de contenedores adaptadas automáticamente a tus necesidades (Ej.: servir una web)?*

Información necesaria para su realización

Ahora que ya sabemos copiar ficheros a un contenedor desde el exterior, es viable usar esta funcionalidad para **hacer provisionamiento automático** con *shell scripts*. Para ello, diseña un *shell script* que instale algunos programas con `apt` o `yum` (dependiendo del tipo de *Linux* que estés usando en el contenedor) y haz un *script* en el host que al ejecutarlo haga lo siguiente:

- Copie el fichero con las instrucciones de provisionamiento del *host* al contendor en una ruta conocida
- Ejecute el fichero copiado para hacer el provisionamiento con la orden vista anteriormente

N3LXD_LIMITES: Actualización de la configuración/límites de una instancia en ejecución *(0,2 puntos)*

Descripción de la actividad

Consiste en **reconfigurar una instancia LXD en funcionamiento**.

Resultados esperados

Eres capaz de actualizar la configuración de una instancia de una imagen LXD mientras está en ejecución, incluyendo añadir un dispositivo de almacenamiento

Información necesaria para su realización

En esta actividad se trata de **actualizar las opciones de una instancia o establecerles límites de uso de recursos mientras esté en funcionamiento** probando diferentes técnicas:

- Se pueden actualizar las opciones de una instancia mientras está en funcionamiento con el comando `lxc config set` así: `lxc config set <instance_name> <option_key>=<option_value> <option_key>=<option_value>…` . Por ejemplo, para cambiar el límite de memoria del contenedor: `lxc config set ubuntu-limited limits.memory 512MiB`
- Se admiten también **establecer otros límites**, por ejemplo de almacenamiento:
  - https://linuxcontainers.org/lxd/docs/master/reference/storage_drivers/#storage-drivers
  - Para ZFS: https://linuxcontainers.org/lxd/docs/master/reference/storage_zfs/
- También se pueden **añadir dispositivos** a las instancias con `lxc config device add`. Necesita el nombre de la instancia, del dispositivo, su tipo y, si es necesario, opciones de este, en función del dispositivo en sí: `lxc config device add <instance_name> <device_name> <device_type> <device_option_key>=<device_option_value> <device_option_key>=<device_option_value> ...` . Por ejemplo, para añadir el almacenamiento que el *host* tiene en `/share/c1` al directorio `/opt` de la instancia: `lxc config device add ubuntu-container disk-storage-device disk source=/share/c1 path=/opt`
- Si lo que se trata es de configurar opciones de un dispositivo de la instancia que ya está añadido a la misma, hay que usar el comando `lxc config device set`. No obstante **no es necesario probar esto último en este ejercicio.**

Nivel 4 (Capitán de navío): Explotación de contenedores de SO

N4LXD_PROFILE: Configuración de contenedores a partir de código: Manejo de perfiles *(0,3 puntos)*

Descripción de la actividad

Consiste en aprender a **crear perfiles de configuración** de instancias de contenedores LXD.

Resultados esperados

Eres capaz de crear un perfil desde cero y modificar partes de este con las funcionalidades de LXD, aplicándoselo a una instancia para probar que funciona correctamente

Información necesaria para su realización

Como se comentó en la teoría**, los perfiles son una forma muy efectiva de configurar una instancia de un contenedor con código** de la forma más adecuada a nuestras necesidades. Esta actividad consiste precisamente en practicar diversas operaciones que nos permitan trabajar con perfiles, siguiendo estas indicaciones.

Información de perfiles disponibles

- **Ver perfiles disponibles**: `lxc profile list`
- **Ver el contenido de un perfil concreto**: `lxc profile show <profile_name>`

En una instalación recién hecha de LXD se crea un perfil `default` que es el que se aplica si no especificamos uno para las instancias que creemos. Este perfil es el siguiente:

```
lxc profile show default
config: {}
description: Default LXD profile
devices:
  eth0:
    name: eth0
    nictype: bridged
    parent: lxdbr0
    type: nic
  root:
    path: /
    pool: zfs
    type: disk
```

```
name: default
used_by:
```

Lo que indica este perfil es que para toda instancia que se cree, **por defecto tendrá una interfaz de red** `eth0` tipo *bridge*, asociada a una tarjeta de red virtual del *host* que se crea cuando instalamos LXD, llamada `lxdbr0`. Además, tendrá un sistema de archivos raíz (`/`) en el *pool* que hemos creado al inicializar `lxd` (`zfs` en nuestro caso) de tipo `disk`. Es importante recordar que **en el momento que especifiquemos un perfil para pasarle a una instancia en su arranque (opción `-p`), el perfil por defecto no se aplicará**, por lo que tenemos dos opciones:

- Incluir el contenido del perfil `default` en el nuevo perfil
- **Añadir a la instancia múltiples perfiles** en su lanzamiento (**se recomienda**). Ej.: `lxc launch images:ubuntu/20.04 ubuntu-profiled -p default -p otroperfil -p otroperfilmas …`

Editar un perfil

El primer paso es crear un perfil vacío que editar: `lxc profile create <profile_name>`. Se puede editar un perfil completo o bien solo cambiar determinadas opciones de instancia de este.

Editar partes de un perfil:

Para cambiar determinadas opciones de instancia se debe usar el comando `lxc profile set`. Necesita el nombre del perfil y los pares clave valor a cambiar: `lxc profile set <profile_name> <option_key>=<option_value> <option_key>=<option_value> ...`

Para añadir un dispositivo suelto a un perfil hay que usar el comando `lxc profile device add`, especificando el nombre del perfil, el del dispositivo, el tipo de dispositivo y opciones de dispositivo si son aplicables: `lxc profile device add <profile_name> <device_name> <device_type> <device_option_key>=<device_option_value> <device_option_key>=<device_option_value> ...`

De la misma forma se pueden configurar opciones de un dispositivo que ya está añadido a un perfil con el comando `lxc profile device set`.

Editar un perfil completo:

Los perfiles no son más que ficheros YAML que se pueden editar con cualquier editor. Normalmente se suelen crear a partir de uno existente. Un perfil típico es el siguiente:

```yaml
name: test-profile
description: "Test profile"
config:
  limits.memory: 2GB
devices:
  test0:
    name: test0
    nictype: bridged
    parent: lxd-my-bridge
    type: nic
```

Las opciones de instancia se añaden como un array en la opción `config`. Los dispositivos de instancia y sus opciones van bajo la opción `devices`. Si tenemos un editor asignado a la terminal podemos usarlo para editar un perfil con: `lxc profile edit <profile_name>`. También podemos crear un fichero YAML con toda la configuración y escribirla completa bajo el nombre de un perfil con: `cat profile.yaml | lxc profile edit <profile_name>`

Con toda esta información se pide **crear un perfil nuevo** para aplicar a una instancia que lancéis **que pruebe vuestra propia configuración**, cambiando algún parámetro de esta que deseéis para probar las funcionalidades mencionadas. Por ejemplo, se puede crear un perfil que juegue con alguno de los límites (`limits`) de los contenedores (https://linuxcontainers.org/lxd/docs/master/instances/), basándose en este:

```
name: limitado
config:
  boot.autostart: "true"
  limits.cpu: "1"
  limits.cpu.priority: "10"
  limits.disk.priority: "10"
  limits.memory: 128MB
  limits.processes: "100"
  description: Un perfil para contenedores limitados en recursos
```

Aplicando perfiles a instancias

Ahora que ya podemos crear perfiles, debemos probar a **aplicar un perfil a una instancia**, probándolo por ejemplo con el perfil anterior. Para ello, seguir las siguientes indicaciones

- Para **aplicar un perfil a una instancia** hay que usar el comando: `lxc profile add <instance_name> <profile_name>`
- Es buena idea **comprobar la configuración** de esta tras añadir la instancia con `lxc config show <instance_name>` y de esta forma comprobar que el nuevo perfil está incluido. Esta opción muestra la configuración del perfil, no de la instancia, lo que quiere decir que **cada vez que se modifica un perfil también se aplican los cambios automáticamente a todas las instancias que lo usan**.
- Para **aplicar uno o más perfiles a una instancia** cuando se lanza hay que usar `--profile` (o `-p`): `lxc launch <image> <instance_name> -p <profile> -p <profile> ...`
- Del mismo modo se pueden **borrar perfiles de una instancia** con: `lxc profile remove <instance_name> <profile_name>`

⚒️ N4LXD_EXPORT: Exportando servicios de contenedores a clientes externos con perfiles *(0,3 puntos)*

🕵️ Descripción de la actividad

Consiste en poder **exportar un servicio que está corriendo dentro de un contenedor LXD al exterior** de este mediante el uso de perfiles y dispositivos tipo *proxy*.

🎖️ Resultados esperados

Eres capaz de crear un perfil que exponga un servicio de un contenedor **en un puerto del *host* de tu elección** y aplicarlo "en caliente" a una instancia existente. Eres capaz de implementarlo exponiendo el servicio de un servidor web del tipo que quieras instalado en un contenedor que crees y arranques previamente. Puedes responder a estas preguntas:

- *¿Por qué crees que este tipo de perfil es mejor aplicarlo en caliente?*

- *¿Qué tendrías que hacer para poner fuera de servicio el servidor momentáneamente de manera sencilla?*

Información necesaria para su realización

Esta actividad consiste en **exponer un puerto al exterior** de forma que uno de los servicios existente, que se está ejecutando en un contenedor en la red privada que se crea para ellos, pueda ser usado desde el exterior. Para ello usaremos un perfil como el de la actividad anterior, que creará un **nuevo dispositivo** tipo `proxy` que mapee un puerto del *host* a uno de la instancia. Hecho esto, adjuntaremos el perfil a una instancia en marcha para que en ese momento un servicio de esta pueda ser ofertado al exterior.

Para ello vamos a crear un perfil `exponer80` como en la actividad anterior, cuyo contenido será un nuevo dispositivo llamado `puerto_80` de tipo `proxy`. Este dispositivo escuchará en `0.0.0.0:80` del *host* (el puerto 80 de todas las direcciones IP asignadas al *host*) y se conectará a la dirección `127.0.0.1:8080` del contenedor. En otras palabras, gracias a este dispositivo **todo acceso al puerto 80 del *host*** se redirigirá (`proxy`) al puerto `8080` del contenedor que tenga este perfil asociado. Esto puede hacerse en línea de comandos con: `lxc profile device add exponer80 puerto_80 proxy connect="tcp:127.0.0.1:8080" listen="tcp:0.0.0.0:80"`

O bien en un fichero `.yaml` como el de la actividad anterior

```
config: {}
description: ""
devices:
  puerto_80:
    connect: tcp:127.0.0.1:8080
    listen: tcp:0.0.0.0:80
    type: proxy
name: exponer80
used_by: []
```

En este caso, en lugar de usar el perfil cuando arranquemos la instancia como en la actividad anterior (que sigue siendo posible) vamos a practicar **adjuntándolo a una que ya está en funcionamiento** y que tenga un servidor web escuchando en el puerto 8080 del contenedor: `lxc profile add <nombre del contenedor> puerto_80`. Podemos ver la configuración aplicada con `lxc config show <nombre del contenedor>` y, por supuesto, quitar el perfil si el servicio debe dejar de ofertarse.

N4LXD_LANCONF: Configuración de redes locales entre contenedores *(0,3 puntos)*

Descripción de la actividad

Consiste en aprender a **crear redes privadas entre contenedores** LXD.

Resultados esperados

Eres capaz de crear una red privada entre dos contenedores a partir de una tarjeta de red del host y configurarles sus IPs para que puedan comunicarse entre ellos.

Información necesaria para su realización

En esta actividad se trata de **crear una red local entre dos contenedores** que se arranquen para este fin, bien de ejercicios anteriores o nuevos. Para ello es necesario que la máquina virtual donde se estén haciendo las pruebas **cree una interfaz de red física** que haga de interfaz de dicha LAN (para *VirtualBox* sirve perfectamente una de red privada), y seguir los pasos vistos en teoría para configurar dicha red. Además, se puede usar este fichero de perfil como guía para hacerlo, **que debe aplicarse conjunto con el perfil** `default` para completar dispositivos y tarjetas de red:

```
config:
  user.network-config: |
    version: 1
    config:
      - type: physical
        name: eth1
        subnets:
          - type: dhcp
            ipv4: true
description: LXD profile with private and public networks
devices:
  eth1:
    name: eth1
    nictype: macvlan
    parent: enp16s0
    type: nic
name: privatepublicnetwork
used_by:
```

Nótese que en este fichero se mencionan **interfaces de red de la instancia** (los que están debajo de la sección `config` (`eth1`, `eth2`, etc.)) y también los del *host* que gestiona los contenedores (los de nuestra máquina virtual). El interfaz que aparece mencionado como `enp16s0` **debe cambiarse** para que **sea el nombre del que realmente tiene el *host*** (tu VM). Fíjate también que el tipo de este segundo interfaz no es `bridged`, sino `macvlan`.

Ambos contenedores deben usar este mismo perfil para poder comunicarse, ya que la IP no es un problema, al asignarse vía DHCP. Para que esto funcione, además hay que tener en cuenta varias cosas:

- La interfaz del host que se use para hacer de interfaz física **debe admitir DHCP**. Podemos usar una red *host-only* de *VirtualBox* para ello, pero **asegurándonos que el servidor DCHP para esta red está activo** en *VirtualBox* (en su configuración), y que dicho interfaz en tu máquina virtual (*host* de LXD) **recibe IPs por DCHP y no tiene una estática**.
- La interfaz `eth1` que aparecerá ahora en todos los contenedores debe configurarse para que acepte IPs vía DHCP, ya que no lo hace por defecto. Para ello hay que editar `/etc/netplan/10-lxc.yaml` de cada contenedor para que ponga esto, haciendo luego un `sudo netplan apply`:

```
network:
  version: 2
  ethernets:
    eth0:
      dhcp4: true
      dhcp-identifier: mac
    eth1:
```

```
      dhcp4: true
```

Hecho esto, ambos contenedores deberían tener una IP compatible entre ellos que les permita comunicarse, que es el objetivo último de esta actividad.

N4LXD_LXDGUI: Aplicaciones con GUIs *(0,2 puntos)*

Descripción de la actividad

Consiste en aprender a **ejecutar aplicaciones con GUI** en contenedores LXD.

Resultados esperados

Eres capaz de configurar un contenedor para que pueda ejecutar aplicaciones gráficas. Puedes contestar a esta pregunta: *¿Qué ventaja crees que tiene ejecutar una aplicación así (por ejemplo, en lo relativo a seguridad)?*

Información necesaria para su realización

Gracias a lo que hemos visto de perfiles, en esta actividad se trata de crear un contenedor donde se instale automáticamente el navegador *Firefox* siguiendo el procedimiento automatizado de provisionamiento que hemos mencionado anteriormente. Hecho esto, con el contenedor arrancado, probar a aplicar este perfil y ver que gracias a él **podemos ejecutar dicho *Firefox* dentro del contenedor**.

```
config:
  environment.DISPLAY: :0
description: Ejecutar aplicaciones graficas X11 en el contenedor
devices:
  x11-socket:
    path: /tmp/.X11-unix/
    source: /tmp/.X11-unix/
    type: disk
name: x11-apps
```

Para probar esto sin problemas, aplicar este perfil a un contenedor cualquiera, instalar una aplicación gráfica (el *Firefox* que mencionamos) y, ya desde el *host*, **ejecutar el comando** `xhost+` para evitar problemas con la redirección del entorno gráfico del contenedor al *host*. Finalmente, lanzar la desde el *host* el contenedor con: `lxc exec mycontainer -- sudo --user ubuntu /usr/bin/firefox`.

NOTA: Si nuestro contenedor ha instalado por ejemplo la GUI completa `xfce4`, se puede lanzar la misma y que se muestre en el *host* con `lxc launch <contenedor> -- sudo -u <usuario> xfce4-panel`

Anexo: Si tienes una GPU dedicada y/o quieres mapear el sonido del host

Esto NO es parte de esta actividad, sino algo que puede ser interesante si queréis crear un contenedor que tenga **aceleración 3D vinculada a vuestra GPU** (en principio marca *NVidia* solamente). Se pueden seguir estas instrucciones: https://blog.simos.info/running-x11-software-in-lxd-containers/ o probar este perfil:


```
config:
  environment.DISPLAY: :0
  nvidia.driver.capabilities: all
  nvidia.runtime: "true"
  raw.idmap: both 1000 1000
  user.user-data: |
    #cloud-config
    runcmd:
      - 'sed -i "s/; enable-shm = yes/enable-shm = no/g" /etc/pulse/client.conf'
      - 'echo export PULSE_SERVER=unix:/tmp/.pulse-native | tee --append
/home/ubuntu/.profile'
    packages:
      - x11-apps
      - mesa-utils
      - pulseaudio
description: LXC profile for ROS
devices:
  PASocket:
    path: /tmp/.pulse-native
    source: /run/user/1000/pulse/native
    type: disk
  X0:
    bind: container
    connect: unix:@/tmp/.X11-unix/X1
    listen: unix:@/tmp/.X11-unix/X0
    security.gid: "1000"
    security.uid: "1000"
    type: proxy
  eth0:
    name: eth0
    network: lxdbr0
    type: nic
  mygpu:
    type: gpu
  root:
    path: /
    pool: default
    type: disk
name: gui
used_by:
```

⚒️ N4LXD_INFRAWEB: Infraestructura para web y base de datos con contenedores *(0,3 puntos)*

🕵️ Descripción de la actividad

Consiste en crear una **infraestructura de contenedores adecuada para servir aplicaciones web** típicas.

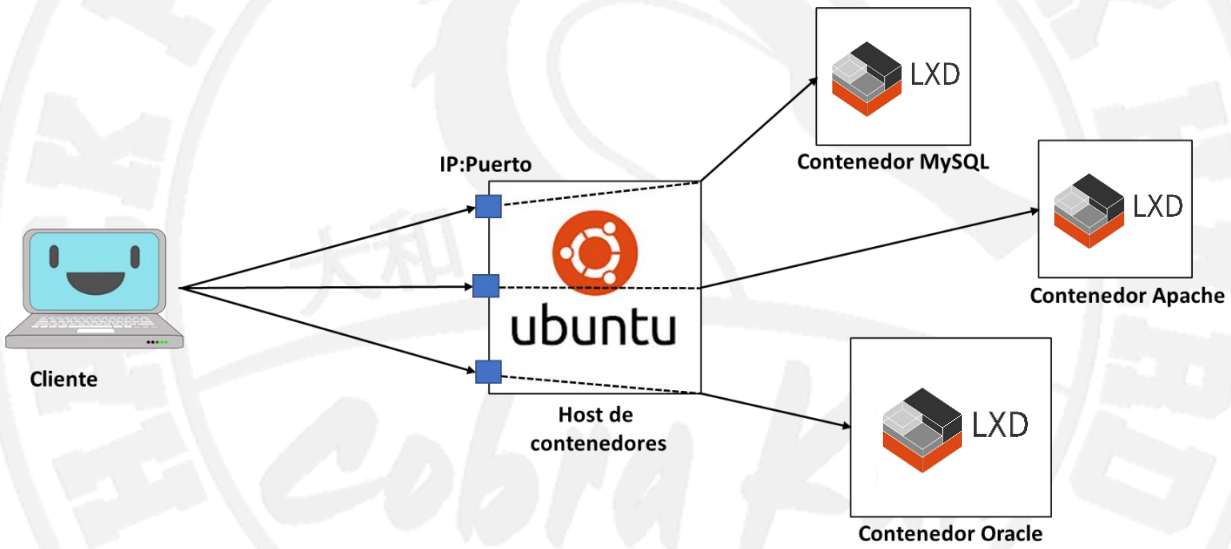
Resultados esperados

Eres capaz de arrancar varios contenedores distintos, configurarlos y **exportar algunos de sus servicios** a través de puertos del host

Información necesaria para su realización

Ahora que sabemos arrancar contenedores y mapear sus servicios a distintos puertos, se pide reproducir la infraestructura de la imagen cumpliendo con los siguientes criterios, siendo estos una extensión de la actividad de nivel 2:

- Contenedor oficial `ubuntu` con `mysql` instalado exportado en el puerto `3306` (puerto por defecto de *MySQL*)
- Contenedor oficial `ubuntu` con `apache` instalado exportando el puerto `8080` (a través del que se accedería al puerto `80` del contenedor)
- Contenedor oficial `ubuntu` en el que se instale un servidor `ssh` que se exponga al exterior a través del puerto `10022` del host



N4LXD_PLATMIW: Instalación de plataformas especializadas de otras asignaturas *(0,5 puntos)*

Descripción de la actividad

Consiste en crear **infraestructuras adecuadas para otras asignaturas** dentro de contenedores LXD.

Resultados esperados

Esta actividad se completará cuando puedas crear satisfactoriamente los contenedores enumerados que preparen automáticamente *software* que se usa en otras asignaturas del máster. Puedes contestar a esta pregunta: *¿Qué ventajas principales ves a hacerlo así frente a usar máquinas virtuales?*

Información necesaria para su realización

En esta actividad se usarán todos los conocimientos anteriores para preparar contenedores de **la última versión estable** de estas herramientas. Estos contenedores corresponderán a entornos de desarrollo que se van a usar en otras asignaturas del máster, por lo que se ruega que **se instalen a partir de imágenes oficiales.**

- **Administración de Sistemas de Persistencia de Objetos**: *Oracle Enterprise Linux 7*
- **Gestores de Contenidos Web**: Un *Ubuntu 18* o superior con XAMPP
- **Programación Orientada a Objetos:** Un *Ubuntu 18* con *Python 3* y `virtualenv`
- **Sistemas de Persistencia de Objetos**: Un *Ubuntu 18* con *Hadoop* (*https://hub.docker.com/r/apache/hadoop*)
- **Sistemas de Seguridad en la Web**: Una máquina *Kali Linux* (versión *Rolling Release*)

Nivel 5 (Almirante): Operaciones avanzadas con contenedores de SO

N5LXD_CLOUDINIT: Escribir un perfil *cloud-init* *(0,2 puntos)*

Descripción de la actividad

Entiendes **qué es un perfil *cloud-init*** y cómo aplicárselo a una instancia LXD.

Resultados esperados

Eres capaz de crear y aplicar un perfil *cloud-init* a una instancia

Información necesaria para su realización

Cloud-init es un perfil especial que permite la preparación de una imagen para ser puesta en marcha con una configuración concreta: https://linuxcontainers.org/lxd/advanced-guide/#cloud-init. Para poder usar un perfil de este tipo se debe crear un nuevo fichero de texto al que llamaremos por ejemplo: `cloud-profile1.profile`. Si lo editamos, tenemos que poner una serie de opciones que se aplicarán sobre la instancia bajo la sección `config`, siguiendo este formato:

```
config:
  user.user-data: |
    #cloud-config
    key: value
```

A modo de ejemplo, mostramos esta plantilla que permite hacer una actualización de paquetes e instalar los paquetes que indiquemos antes de arrancar la instancia de un contenedor.

```
config:
  user.user-data: |
    #cloud-config
    package_upgrade: true
    packages:
      - <paquete 1>
      - <paquete 2>
      …
      - <paquete N>
```

Con esta información se pide crear un perfil *cloud-init* para una instancia nueva y existente y aplicarlo para comprobar que funciona correctamente.

N5LXD_CUSTOMLXD: Creación de una imagen personalizada de LXD *(0,4 puntos)*

Descripción de la actividad

Consiste en **crear una imagen LXD personalizada** para luego hacer instancias a partir de ella.

Resultados esperados

Puedes crear una imagen de LXD personalizada con el software que desees.

Información necesaria para su realización

En esta actividad se debe seguir este manual de procedimientos https://ubuntu.com/tutorials/create-custom-lxd-images#1-overview para **crear una imagen personalizada de LXD que tenga *Apache* preinstalado**, procediendo luego a crear un contenedor de esta para comprobar que el procedimiento es correcto. No se debe publicar la misma, ya que no es necesario.

N5LXD_DOCKERLXD: *Docker y LXD* *(0,2 puntos)*

Descripción de la actividad

Consiste en **poder lanzar contenedores de aplicaciones *Docker*** dentro de un contenedor LXD. **Requiere tener un Nivel 1 de *Docker* o superior**.

Resultados esperados

Eres capaz de crear una imagen LXD con el software que soporta contenedores *Docker* instalado, uniendo así ambos módulos de la asignatura para conseguir un fin. Puedes contestar a esta pregunta: *¿Cómo usarías ambas tecnologías de contenedores en tándem?*

Información necesaria para su realización

(Esta actividad es opcional para completar este nivel si no has hecho el módulo de Docker hasta el nivel 1 o superior)

En esta actividad vamos a **combinar los dos tipos de tecnología de contenedores** en una haciendo que un contenedor LXD de tipo *Ubuntu* como el que hemos venido usando hasta el momento pueda albergar contenedores *Docker* del estilo a los que se crean en el otro bloque de ejercicios, combinando de forma efectiva el uso de ambos con los conocimientos adquiridos hasta el momento. Para ello es necesario usar este fichero de perfil para configurar la imagen de LXD usada:

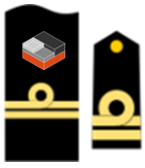
```
config:
  raw.lxc: |-
    lxc.cgroup.devices.allow = a
    lxc.mount.auto = proc:rw sys:rw cgroup:rw
```

```
    lxc.apparmor.profile = unconfined
    lxc.cap.drop =
  security.nesting: "true"
  security.privileged: "true"
description: ""
devices: {}
name: docker
used_by:
```

Y finalmente instalar dentro del contendor los paquetes `docker` y `docker.io`. Hecho esto, crear un contenedor *Docker* básico dentro del contenedor LXD y comprobar que se puede acceder a él o a sus servicios.

Insignias y Autoevaluación

Esta asignatura hace uso de insignias. Esto quiere decir que, cuando vayas desbloqueando niveles de maestría en la tecnología, podrás obtener **una insignia pública** que luego podrás usar en tu currículum gracias al soporte de estas del campus virtual. Este enlace te enseña a usar esta característica para poder unirlas a tu currículum: https://docs.moodle.org/all/es/Insignias. Cada insignia pública **lleva adjuntos los textos de las habilidades que desbloquea** que puedes consultar en la siguiente tabla, para que cualquiera los vea si decides usarlas y sepa lo que has conseguido hacer, validado por mí. De esta manera, mi intención es ayudarte a aumentar tus posibilidades de contratación en un futuro, **exponiendo claramente lo que has podido hacer en esta asignatura** referente a cada tecnología sobre la que has decidido evaluarte.

Rango de Maestría en la Tecnología y Habilidades Demostradas		Insignia Pública Obtenida
Nivel 1: Cabo de Contenedores de SO		
	Puede instalar LXD y tener un sistema de contenedores de SO en funcionamiento	
	Puede arrancar un contenedor preconstruido	
	Puede conectarse a un contenedor, interactuar con él, pararlo y borrarlo cuando ya no es necesario ***(2 ejercicios)***	
	Puede crear un contenedor con un servidor web	
Nivel 2: Sargento de Contenedores de SO		
	Sabe cómo gestionar y encontrar instancias de contenedores en funcionamiento	
	Puede arrancar e instalar en un contenedor cualquier software que necesite para una función concreta	
	Puede conectarse vía SSH a un contenedor	
	Puede crear una infraestructura de web y base de datos con contenedores	
Nivel 3: Teniente de Navío de Contenedores de SO		
	Sabe cómo modificar la configuración de un contenedor en el arranque y durante su ejecución	
	Sabe cómo extraer y copiar ficheros de / a un contenedor cualquiera	
	Puede crear un volumen de almacenamiento adicional a un contenedor	
	Puede automatizar el provisionamiento de contenedores	
	Puede actualizar la configuración y límites de una instancia en tiempo de ejecución	
Nivel 4: Capitán de Navío de Contenedores de SO		
	Puede usar perfiles para configurar contenedores de forma más sencilla	

	Puede exponer un servicio que reside en un contenedor al exterior de su *host*	
	Puede crear redes privadas entre contenedores	
	Sabe cómo arrancar aplicaciones que usen GUI que están instaladas dentro de un contenedor	
	Puede instalar plataformas y software útiles en varios contextos dentro de los contenedores	
Nivel 5: Almirante de Contenedores de SO		
	Es capaz de crear y usar un perfil *cloud-init*	
	Sabe crear una imagen personalizada de LXD	
	Puede crear un contenedor de SO que dentro pueda albergar contenedores de aplicaciones	
	**Cruz con el distintivo azul en Contenedores de SO**: Ha completado con éxito **4 puntos** (aproximadamente el 80%) de las tareas correspondientes de los 5 niveles de la parte de *Contenedores de SO*, demostrando así su maestría en esta tecnología y estar preparado para usarla en tu futuro trabajo de forma efectiva