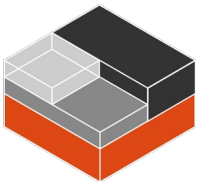
Centro de Procesamiento de Datos



Práctica 6. LXD y despliegue aplicaciones con JUJU / LXD

LXD permite ejecutar procesos y mantener recursos aislados de forma eficiente y compartiendo el núcleo del sistema operativo. Los dispositivos están virtualizados de forma que cada contenedor ve sus propios dispositivos. Los procesos que se ejecutan son nativos del equipo host y no es necesario crear un hardware virtualizado completo, por lo que el funcionamiento global es más eficiente.

Aunque se hace referencia a LXD como contenedores, tienen un funcionamiento distinto a Docker. LXD virtualiza una máquina con sus interfaces, Docker está mas orientado a aplicaciones.

JUJU permite crear fácilmente contenedores con aplicaciones preinstaladas y una configuración básica para evaluarlas y administrarlas.

Deben presentarse en un documento pdf, con la siguiente información:
-url de la dirección *.onion donde está activa la página wiki
-wiki modificado donde aparece el nombre del alumno
-captura (personalizada) de la página wiki desde el TBB (Tor Browser Bundle)
-captura (personalizada) del acceso por ssh al servidor utilizando la red tor
-(opcional) captura de gitlab desde desde el TBB (Tor Browser Bundle)
-(opcional) captura del mysql con replicación

Desarrollo:

LXD funciona preferentemente en Ubuntu, ya que está desarrollado por Canonical. También puede funcionar en CentOS.

Si no tenemos Ubuntu, podemos instalarlo con Vagrant, ya que LXD puede instalarse sobre máquinas virtuales. Ubuntu 20.04 LTS, box: ubuntu/focal64

```
vagrant init ubuntu/focal64
vagrant up
vagrant ssh
```

Una vez dentro de Ubuntu instalamos ZFS que ofrece mejor soporte de ficheros para LXD.

```
sudo apt install zfsutils-linux
```

Iniciamos el servicio, confirmando todos los pasos por defecto

```
sudo lxd init
```

Primeros pasos con contenedores LXD

En el caso del contenedor con vagrant, añadir el usuario vagrant al grupo lxd, o bien añadir sudo en las siguientes órdenes. Para añadir el usuario:

```
sudo usermod -a -G lxd vagrant
```

Es necesario salir de la shell y volver a entrar.

Creamos un contenedor ubuntu que llamaremos c0:

```
lxc launch ubuntu: c0
```

Podemos comprobar que está funcionando

```
lxc list
```

Ver qué imágenes de contenedor tenemos

```
lxc image list
```

Accedemos mediante shell al contenedor

```
lxc exec c0 bash
```

Comprobamos la versión de Ubuntu

```
lsb_release -a
```

Copiamos el contenedor creando uno nuevo

```
lxc copy c0 c1
```

Comprobamos que se ha creado

```
lxc list
```

Activamos el contedor

```
lxc start c1
```

Accedemos al segundo contenedores

```
lxc exec c1 bash
```

Paramos y borramos el segundo contenedor

```
lxc stop c1
lxc delete c1
```

Mostrar la lista de imágenes disponibles

```
lxc image list images:
```

Existen 3 repositorios con imágenes:
ubuntu: (para imágenes estables de Ubuntu)
ubuntu-daily: (para imágenes diarias de Ubuntu)
images: (para un conjunto de otras distribuciones)

Lista de contenedores

```
lxc list
lxc image list images:
```

Ya que LXD está orientado a definir contenedores con un sistema operativo, podemos encontrar: ubuntu, debian, CentOS, Oracle, gentoo, plamo, Alpine, ...

Creamos un contenedor basado en Centos 7 y que se llame c2

```
lxc launch images:centos/7 c2
```

Accedemos mediante shell al contenedor c2

```
lxc exec c2 bash
```

Comprobamos la versión de Centos

```
More /etc/redhat-release
```

Comprobación de la red
Activamos 2 contenedores y comprobamos el acceso entre ambos verificando la ip asignada a cada contenedor

```
lxc list
```

Compartiendo un directorio local con el contenedores

```
mkdir disco1
lxc config device add c1 disco1 disk source=/home/usuario/prueba1 path=/mnt/disco1
```

Si queremos acceso de lectura y escritura hay que comprobar los permisos del propietario del contenedor

```
sudo ls -l /var/lib/lxd/containers
```

Limitando recursos en un contenedor

```
lxc config set c1 limits.memory 512MB
lxc config show c1
```

Podemos comprobar con lxc exec c1 /bin/bash
free -m

Algunos parámetros:
boot.autostart
limits.cpu
limits.cpu.allowance
limits.memory

Operaciones con contenedores LXD

Creando un contenedor
Los contenedores se crean a partir de imágenes y según donde están: remotos, built-in y locales

Órdenes básicas

Creamos un contenedor:

```
#la última versión
lxc launch ubuntu: c0

#versión específica
lxc launch ubuntu:16.04 c1

#versión 32 bits
lxc launch ubuntu:14.04/i386 c2

#versión de alpine linux (ligera y segura)
lxc launch images:alpine/3.8 c2a

#versión centos
lxc launch images:centos/7 mi_centos1
#versiones no testeadas
lxc launch ubuntu-daily:devel c3
```

Parar un contenedores

```
lxc stop c1
#parada forzada
lxc stop c1 --force
```

Iniciamos un contenedor

```
lxc start c1
```

Reiniciar

```
lxc restart c1
lxc restart c1 --force
```

Pausar

```
lxc pause c1
```

Borrar

```
lxc delete c1
```

Copiar un contenedor

```
lxc copy <contenedor_origen> <contenedor_destino>
```

Mover un contenedor

```
lxc move
```

Información

```
lxc info c1
```

Executamos un shell

```
lxc exec c1 bash
```

Leer un fichero del contenedor

```
lxc file pull <contenedor>/<path> <dest>
```

Grabar un fichero en el contenedor

```
lxc file push <source> <contenedor>/<path>
```

Editar un fichero

```
lxc file edit <contenedor>/<path>
```

Snapshots:

```
lxc snapshot <contenedor> <nombre_contenedor>
```

Restaurar

```
lxc restore <contenedor> <nombre_contenedor>
```

Renombrar

```
lxc move <contenedor>/<nombre_contenedor> <contenedor>/<nuevo_nombre_contenedor>
```

Copiar un contenedor desde un snapshot

```
lxc copy <contenedor>/<nombre_contenedor> <nuevo contenedor>
```

Borrar snapshot

```
lxc delete <contenedor>/<nombre_contenedor>
```

Importar

```
lxc image import <fichero> --alias mi_alias
#después se puede utilizar dicho alias
lxc launch mi_alias <contenedor>
```

Publicar

```
lxc publish <contenedor> --alias nombre
```

Interfaz en modo bridge

```
lxc config device add c1 eth1 nic nictype=bridged parent=lxdbr0
```

Accediendo remotamente a los contenedores

Creamos las claves RSA y utilizamos ssh-copy-id

II) Despliegue aplicaciones con JUJU / LXD

Recordamos

```
Creación de un contenedor: lxc launch ubuntu:xenial c1
Mostrar contenedores: lxc list
Mostrar imágenes: lxc image list
Ejecutar en el contenedor (c1): lxc exec c1 bash
```

Si queremos borrar por completo el almacenamiento de lxd:

```
sudo service lxd stop
rm -rf varlib/lxd
sudo service lxd start
```

III) Instalación de JUJU

Para instalarlo en nuestro equipo con snap

```
sudo snap install juju --classic
```

La documentación de Juju está disponible en : https://docs.jujucharms.com/2.4/en/

IV) Creación del controlador

Una vez instalador definimos un controlador. Los controladores establecen los recursos que se van a utilizar para desplegar los contenedores.
En nuestro caso estamos utilizando contenedores locales basados en LXD, por lo que podemos crear nuestro contenedor con:

```
juju bootstrap localhost lxd-test
```

Se crea un contendor básico con los recursos y servicios básico para que Juju pueda instalar aplicaciones. Este proceso puede durar 1-2 minutos.

Para mostrar la lista de controladores :

```
juju controllers
```

V) Despliegue de aplicaciones

Una vez disponemos de un controlador podemos crear contenedores con aplicaciones. Veamos un ejemplo:

```
juju deploy cs:bundle/wiki-simple
```

Comprobamos su funcionamiento con :

```
juju status
```

Y comprobamos en qué máquina se está ejecutando wiki:

```
Model    Controller  Cloud/Region        Version  SLA
default  lxd-test    localhost/localhost  2.2.4    unsupported

App    Version  Status   Scale  Charm      Store       Rev  OS      Notes
mysql           unknown      1  mysql      jujucharms   55  ubuntu
wiki            unknown      1  mediawiki  jujucharms    5  ubuntu

Unit      Workload  Agent  Machine  Public address  Ports     Message
mysql/0*  unknown   idle   0        10.95.56.64     3306/tcp
wiki/0*   unknown   idle   1        10.95.56.170    80/tcp

Machine  State    DNS           Inst id        Series  AZ  Message
0        started  10.95.56.64   juju-cdb1f3-0  trusty      Running
1        started  10.95.56.170  juju-cdb1f3-1  trusty      Running

Relation provider  Requirer      Interface  Type
mysql:cluster      mysql:cluster  mysql-ha   peer
mysql:db           wiki:db        mysql      regular
```

Abrimos una conexión desde nuestro ordenador con un túnel SSH.
Editamos el fichero *config* del directorio ~/.ssh

(O hacer como se ve más adelante con el comando ssh)

.ssh/config (ej. para el usuario **7**)

```
Host abfs
  hostname abfs.ugr.es
  Port 3307
  User cpd7
  LocalForward 8080 10.95.56.170:80
```

Indicamos la IP del servidor wiki

Esto abre un túnel desde el puerto 8080 local al puerto 80 de la máquina que ejecuta el wiki.

Podemos abrir un navegador para acceder a la configuración

```
http://localhost:8080
```

La lista de charms y bundles disponibles está en: https://jujucharms.com/store

Si queremos eliminar las aplicaciones de juju:

```
juju remove-application wiki --destroy-storage
juju remove-application mysql --destroy-storage
```

Tras unos segundos desaparecen los recursos utilizados por las aplicaciones (comprobar con *juju status*).

VI) Utilizar la red TOR para acceder directamente al contenedor

La red TOR permite crear servicios ocultos y acceder de forma transparente a dicho servidor, ya sean nodos, máquinas virtuales o contenedores.
Configurando el servidor

instalamos tor en el servidor:

```
apt install tor
```

Editamos /etc/tor/torrc

```
RunAsDaemon 1
HiddenServiceDir /root/servicio1
HiddenServicePort 22 127.0.0.1:22
HiddenServicePort 80 10.95.56.170:80
```

Indicamos la IP del servidor wiki

sudo mkdir /root/servicio1

Iniciamos con:

```
tor
```

Esta orden inicia el acceso a la red TOR y permite el acceso a los puertos definidos.
El directorio que utilizamos para definir el servicio debe tener permisos de grupo y otros desactivado:

```
chmod go-rwx /~~etc/~~servicio1
```

/root/

En el directorio /root/servicio1 se crean dos ficheros:
hostname: Contiene el nombre del servidor para serutilizado por los clientes: Ej. i549xykbkyeupam4.onion
private_key: Clave privada que permite identificar al servidor

Configurando el cliente

Instalamos el TBB (Tor Browser Bundle)
Iniciamos el TBB

```
./start-tor-browser
```

Se puede acceder directamente utilizando la dirección que aparecen el fichero *hostname* del directorio definido en el servidor.

Para acceder mediante ssh, añadimos las siguientes líneas el el fichero .ssh/config que editamos anteriormente en nuestro equipo (ojo: estas líneas NO se incluyen dentro del contenedor).

En tu ordenador host

```
Host *.onion
  Port 22
  ProxyCommand nc -X 5 -x localhost:9150 %h %p
```

Para acceder remotamente en la máquina vagrant:
Modificar */etc/ssh/sshd_config*

```
PasswordAuthentication yes
```

Reiniciar sshd

```
sudo systemctl restart ssh.service
```

Creamos tunel con:
ssh -p 2222 vagrant@localhost -L5080:10.249.109.187:80

y ahora podemos acceder al navegador con http://localhost:5080/

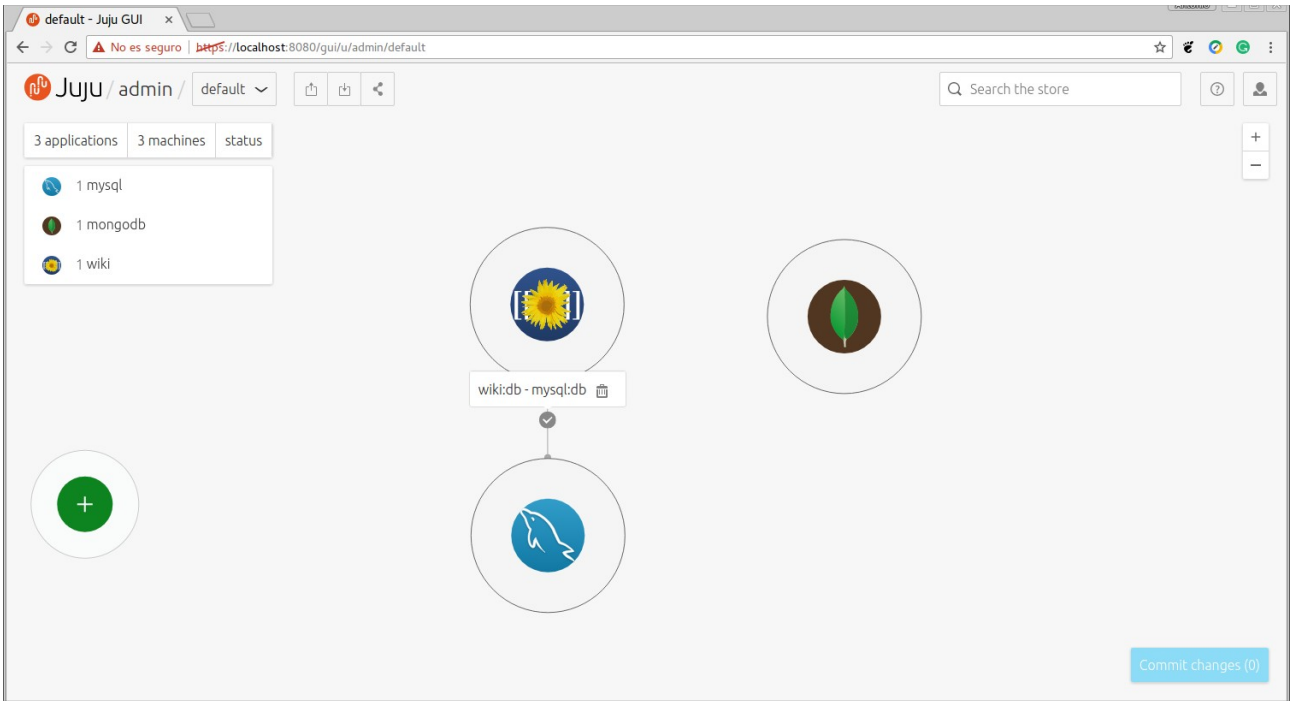
VII) interfaz GUI para Juju

iniciamos el entorno GUI con:

```
juju gui
```

https://10.95.57.36:15070/gui/u/admin/default
Your login credential is:
username: admin
password: **********

Creamos un túnel ssh para acceder al entorno:



Podemos añadir *mongodb* mediante el entorno gui.

VIII) Instalación de otras aplicaciones:
Procedemos a instalar otras aplicaciones como gitlab:

```
juju deploy cs:~spiculecharms/trusty/gitlab-0
juju expose gitlab
```

O gitlab server:

```
juju deploy cs:~spiculecharms/gitlab-server-8
juju expose gitlab
```

IX) Instalación de Mysql con replicación

MySQL permite replicar bases de datos mediante nuevas instancias. Esto permite, por ejemplo, balancear la carga de consultas entre múltiples instancias o utilizar otra instancia para realizar copias de seguridad, todo ello sin afectar al funcionamiento de la instancia principal.

Seguir las indicaciones de : https://jujucharms.com/mysql/