

Práctica 9: Virtualización con LXD y gestión de alertas con Telegram

Autor: Manuel Díaz-Meco Terrés

Fecha: 10 de diciembre 2024

Introducción

En esta práctica se hará uso de LXD para crear un sistema de monitorización de múltiples nodos, utilizando Telegram como sistema rápido para aviso de alertas.

Creación del entorno y contenedores

Lanzamos el comando `vagrant init ubuntu/jammy64` para crear el entorno correspondiente. Al hacer esto se nos creará nuestro *Vagrantfile* que tendremos que modificar como se indica en el guión. Yo he optado por usar la ip `192.168.56.10` ya que la que viene en el guión, `192.168.33.10`, no es aceptada.

```
> cd ~/Universidad/5/CPD/pr9
> vagrant init ubuntu/jammy64
A `Vagrantfile` has been placed in this directory. You are now
ready to `vagrant up` your first virtual environment! Please read
the comments in the Vagrantfile as well as documentation on
`vagrantup.com` for more information on using Vagrant.
> nano Vagrantfile
> vagrant up
Bringing machine 'default' up with 'virtualbox' provider...
==> default: Importing base box 'ubuntu/jammy64'...
==> default: Matching MAC address for NAT networking...
==> default: Checking if box 'ubuntu/jammy64' version '20241002.0.0' is up to date...
==> default: Setting the name of the VM: pr9_default_1733820919179_72760
==> default: Clearing any previously set network interfaces...
The IP address configured for the host-only network is not within the
allowed ranges. Please update the address used to be within the allowed
ranges and run the command again.

Address: 192.168.33.10
Ranges: 192.168.56.0/21

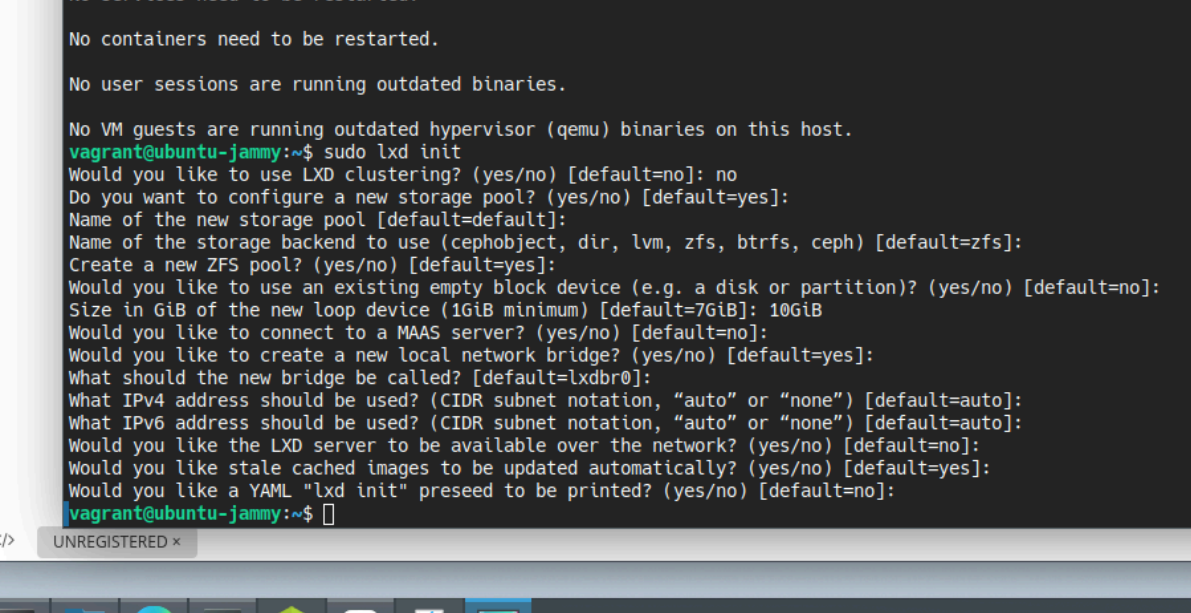
Valid ranges can be modified in the /etc/vbox/networks.conf file. For
more information including valid format see:

https://www.virtualbox.org/manual/ch06.html#network_hostonly
> nano Vagrantfile
> vagrant up
Bringing machine 'default' up with 'virtualbox' provider...
==> default: Checking if box 'ubuntu/jammy64' version '20241002.0.0' is up to date...
==> default: Clearing any previously set network interfaces...
==> default: Preparing network interfaces based on configuration...
default: Adapter 1: nat
default: Adapter 2: hostonly
==> default: Forwarding ports...
default: 22 (guest) => 2222 (host) (adapter 1)
==> default: Running 'pre-boot' VM customizations...
==> default: Booting VM...
==> default: Waiting for machine to boot. This may take a few minutes...
default: SSH address: 127.0.0.1:2222
default: SSH username: vagrant
default: SSH auth method: private key
```

Ejecutando `vagrant ssh` estaremos dentro de la máquina virtual creada. A continuación ejecutamos:

```
sudo apt update
sudo apt -y install zfsutils-linux
sudo lxd init
```

Las opciones seleccionadas del último comando son las siguientes:



```
No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
vagrant@ubuntu-jammy:~$ sudo lxd init
Would you like to use LXD clustering? (yes/no) [default=no]: no
Do you want to configure a new storage pool? (yes/no) [default=yes]:
Name of the new storage pool [default=default]:
Name of the storage backend to use (cephobject, dir, lvm, zfs, btrfs, ceph) [default=zfs]:
Create a new ZFS pool? (yes/no) [default=yes]:
Would you like to use an existing empty block device (e.g. a disk or partition)? (yes/no) [default=no]:
Size in GiB of the new loop device (1GiB minimum) [default=7GiB]: 10GiB
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 the LXD server 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]:
vagrant@ubuntu-jammy:~$
```

Añadimos el usuario al grupo `lxd` con `sudo usermod -a -G lxd vagrant` y salimos y volvemos a entrar para que se hagan efectivos los cambios. Ahora creamos un contenedor ubuntu, probamos que funciona y ver las imágenes del contenedor con los comandos:

```
lxc launch ubuntu: c0
lxc list
lxc image list
```

```
pr9 : bash — Konsole
File Edit View Bookmarks Plugins Settings Help
Would you like the LXD server 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]:
vagrant@ubuntu-jammy:~$ sudo usermod -a -G lxd vagrant
vagrant@ubuntu-jammy:~$ exit
logout
> vagrant ssh
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 5.15.0-122-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

System information as of Tue Dec 10 09:09:04 UTC 2024

System load:  0.02          Processes:      159
Usage of /:   4.3% of 38.7GB Users logged in: 0
Memory usage: 7%           IPv4 address for enp0s3: 10.0.2.15
Swap usage:  0%

Expanded Security Maintenance for Applications is not enabled.

35 updates can be applied immediately.
27 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

New release '24.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Tue Dec 10 09:01:03 2024 from 10.0.2.2
vagrant@ubuntu-jammy:~$ lxc launch ubuntu: c0
Creating c0
Starting c0
vagrant@ubuntu-jammy:~$ lxc list
+-----+-----+-----+-----+-----+-----+
| NAME | STATE | IPV4 | IPV6 | TYPE | SNAPSHOTS |
+-----+-----+-----+-----+-----+-----+
| c0   | RUNNING | 10.75.48.207 (eth0) | fd42:e794:4e5f:b296:216:3eff:fee1:e3e7 (eth0) | CONTAINER | 0 |
+-----+-----+-----+-----+-----+-----+
vagrant@ubuntu-jammy:~$ lxc image list
+-----+-----+-----+-----+-----+-----+-----+-----+
| ALIAS | FINGERPRINT | PUBLIC | DESCRIPTION | ARCHITECTURE | TYPE | SIZE | UPLOAD DATE |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 40d8df642812 | no | ubuntu 24.04 LTS amd64 (release) (20241119) | x86_64 | CONTAINER | 254.20MiB | Dec 10, 2024 at 9:10am (UTC) |
+-----+-----+-----+-----+-----+-----+-----+-----+
vagrant@ubuntu-jammy:~$
```

Añadimos mediante shell al contenedor y comprobamos la última versión de ubuntu:

```
+-----+-----+-----+-----+-----+-----+
| 40d8df642812 | no | ubuntu 24.04 LTS amd64 (r
+-----+-----+-----+-----+-----+-----+
vagrant@ubuntu-jammy:~$ lxc exec c0 bash
root@c0:~# lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 24.04.1 LTS
Release:        24.04
Codename:       noble
root@c0:~#
```

Ahora copiamos 'duplicamos' este contenedor y lo llamamos c1. Hacemos lo mismo que antes, solo que el contenedor está parado en el momento de su creación, por lo que hay que arrancarlo. Tras esto lo paramos y borramos y vemos el resultado:

```
root@c0:~# exit
exit
vagrant@ubuntu-jammy:~$ lxc copy c0 c1
vagrant@ubuntu-jammy:~$ lxc list
+-----+-----+-----+-----+-----+-----+
| NAME | STATE | IPV4 | IPV6 | TYPE | SNAPSHOTS |
+-----+-----+-----+-----+-----+-----+
| c0 | RUNNING | 10.75.48.207 (eth0) | fd42:e794:4e5f:b296:216:3eff:fee1:e3e7 (eth0) | CONTAINER | 0 |
+-----+-----+-----+-----+-----+-----+
| c1 | STOPPED | | | CONTAINER | 0 |
+-----+-----+-----+-----+-----+-----+

vagrant@ubuntu-jammy:~$ lxc start c1
vagrant@ubuntu-jammy:~$ lxc list
+-----+-----+-----+-----+-----+-----+
| NAME | STATE | IPV4 | IPV6 | TYPE | SNAPSHOTS |
+-----+-----+-----+-----+-----+-----+
| c0 | RUNNING | 10.75.48.207 (eth0) | fd42:e794:4e5f:b296:216:3eff:fee1:e3e7 (eth0) | CONTAINER | 0 |
+-----+-----+-----+-----+-----+-----+
| c1 | RUNNING | 10.75.48.247 (eth0) | | CONTAINER | 0 |
+-----+-----+-----+-----+-----+-----+

vagrant@ubuntu-jammy:~$ lxc exec c1 bash
root@c1:~# exit
exit
vagrant@ubuntu-jammy:~$ lxc stop c1
vagrant@ubuntu-jammy:~$ lxc delete c1
vagrant@ubuntu-jammy:~$ lxc list
+-----+-----+-----+-----+-----+-----+
| NAME | STATE | IPV4 | IPV6 | TYPE | SNAPSHOTS |
+-----+-----+-----+-----+-----+-----+
| c0 | RUNNING | 10.75.48.207 (eth0) | fd42:e794:4e5f:b296:216:3eff:fee1:e3e7 (eth0) | CONTAINER | 0 |
+-----+-----+-----+-----+-----+-----+

vagrant@ubuntu-jammy:~$
```

Parece ser que el servidor de *images* por defecto ya no está vacío, como se indica en el guión. Por ello, procedemos sin utilizar el otro repositorio que se indica. Ejecutamos:

```
lxc remote list
lxc image list images: | head
lxc image list images: 'alpine'
```

De esta forma podemos ver las imágenes que hay disponibles o consular las que hay con una palabra clave, como vemos en el tercer comando.

```
vagrant@ubuntu-jammy:~$ lxc remote list
+-----+-----+-----+-----+-----+-----+-----+
| NAME | URL | PROTOCOL | AUTH TYPE | PUBLIC | STATIC | GLOBAL |
+-----+-----+-----+-----+-----+-----+-----+
| images | https://images.lxd.canonical.com | simplestreams | none | YES | NO | NO |
+-----+-----+-----+-----+-----+-----+-----+
| local (current) | unix:// | lxd | file access | NO | YES | NO |
+-----+-----+-----+-----+-----+-----+-----+
| ubuntu | https://cloud-images.ubuntu.com/releases | simplestreams | none | YES | YES | NO |
+-----+-----+-----+-----+-----+-----+-----+
| ubuntu-daily | https://cloud-images.ubuntu.com/daily | simplestreams | none | YES | YES | NO |
+-----+-----+-----+-----+-----+-----+-----+
| ubuntu-minimal | https://cloud-images.ubuntu.com/minimal/releases/ | simplestreams | none | YES | YES | NO |
+-----+-----+-----+-----+-----+-----+-----+
| ubuntu-minimal-daily | https://cloud-images.ubuntu.com/minimal/daily/ | simplestreams | none | YES | YES | NO |
+-----+-----+-----+-----+-----+-----+-----+

vagrant@ubuntu-jammy:~$ lxc image list images: | head
+-----+-----+-----+-----+-----+-----+-----+
| ALIAS | FINGERPRINT | PUBLIC | DESCRIPTION | ARCHITECTURE | TYPE | SIZE | UPLOAD DATE |
+-----+-----+-----+-----+-----+-----+-----+
| almalinux/8 (3 more) | 614e44fbc5f4 | yes | AlmaLinux 8 amd64 (20241210_0014) | x86_64 | VIRTUAL-MACHINE | 884.52MiB | Dec 10, 2024 at 12:00am (UTC) |
+-----+-----+-----+-----+-----+-----+-----+
| almalinux/8 (3 more) | d2522c7f8014 | yes | AlmaLinux 8 amd64 (20241210_0014) | x86_64 | CONTAINER | 129.00MiB | Dec 10, 2024 at 12:00am (UTC) |
+-----+-----+-----+-----+-----+-----+-----+
| almalinux/8/arm64 (1 more) | 23653ce8c0a0 | yes | AlmaLinux 8 arm64 (20241209_0035) | aarch64 | CONTAINER | 125.22MiB | Dec 9, 2024 at 12:00am (UTC) |
+-----+-----+-----+-----+-----+-----+-----+
| almalinux/8/cloud (1 more) | 0b34f5dd15d8 | yes | AlmaLinux 8 amd64 (20241210_0014) | x86_64 | CONTAINER | 148.64MiB | Dec 10, 2024 at 12:00am (UTC) |
+-----+-----+-----+-----+-----+-----+-----+

vagrant@ubuntu-jammy:~$ lxc image list images: 'alpine'
+-----+-----+-----+-----+-----+-----+-----+
| ALIAS | FINGERPRINT | PUBLIC | DESCRIPTION | ARCHITECTURE | TYPE | SIZE | UPLOAD DATE |
+-----+-----+-----+-----+-----+-----+-----+
| alpine/3.17 (3 more) | 8c43f21e77f4 | yes | Alpine 3.17 amd64 (20241210_0022) | x86_64 | VIRTUAL-MACHINE | 108.00MiB | Dec 10, 2024 at 12:00am (UTC) |
+-----+-----+-----+-----+-----+-----+-----+
| alpine/3.17 (3 more) | 630415872b46 | yes | Alpine 3.17 amd64 (20241210_0022) | x86_64 | CONTAINER | 2.94MiB | Dec 10, 2024 at 12:00am (UTC) |
+-----+-----+-----+-----+-----+-----+-----+
| alpine/3.17/arm64 (1 more) | 59285cc8425d | yes | Alpine 3.17 arm64 (20241209_0020) | aarch64 | CONTAINER | 2.71MiB | Dec 9, 2024 at 12:00am (UTC) |
+-----+-----+-----+-----+-----+-----+-----+
| alpine/3.17/cloud (1 more) | 34afd52b8da8 | yes | Alpine 3.17 amd64 (20241210_0023) | x86_64 | VIRTUAL-MACHINE | 135.88MiB | Dec 10, 2024 at 12:00am (UTC) |
+-----+-----+-----+-----+-----+-----+-----+
```

Creamos ahora un contenedor basado en *CentOS9* llamado *c2*, ya que LXD está orientado a definir contenedores con un sistema operativo:

```
lxc launch images:centos/9-Stream c2
lxc exec c2 bash
[root@c2 ~]# more /etc/redhat-release
```

Con el último comando se comprueba la versión del sistema operativo.

```
| alpine/edge/cloud/arm64 | a/4815/ff562 | yes | Alpine edge a
+-----+-----+-----+-----+
vagrant@ubuntu-jammy:~$ lxc launch images:centos/9-Stream c2
Creating c2
Starting c2
vagrant@ubuntu-jammy:~$ lxc exec c2 bash
[root@c2 ~]# more /etc/redhat-release
CentOS Stream release 9
[root@c2 ~]# exit
exit
vagrant@ubuntu-jammy:~$
```

Para comprobar la conexión entre ambos contenedores debemos ver su ip y conectarnos desde uno de ellos al otro mediante el uso de `ping`:

```
vagrant@ubuntu-jammy:~$ lxc list
+-----+-----+-----+-----+-----+-----+
| NAME | STATE | IPV4 | IPV6 | TYPE | SNAPSHOTS |
+-----+-----+-----+-----+-----+-----+
| c0 | RUNNING | 10.75.48.207 (eth0) | fd42:e794:4e5f:b296:216:3eff:fee1:e3e7 (eth0) | CONTAINER | 0 |
+-----+-----+-----+-----+-----+-----+
| c2 | RUNNING | 10.75.48.118 (eth0) | fd42:e794:4e5f:b296:9638:7d0e:a242:59 (eth0) | CONTAINER | 0 |
+-----+-----+-----+-----+-----+-----+

vagrant@ubuntu-jammy:~$ lxc exec c2 bash
[root@c2 ~]# ping 10.75.48.207
PING 10.75.48.207 (10.75.48.207) 56(84) bytes of data:
64 bytes from 10.75.48.207: icmp_seq=1 ttl=64 time=0.296 ms
64 bytes from 10.75.48.207: icmp_seq=2 ttl=64 time=0.158 ms
64 bytes from 10.75.48.207: icmp_seq=3 ttl=64 time=0.146 ms
^C
--- 10.75.48.207 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2005ms
rtt min/avg/max/mdev = 0.146/0.200/0.296/0.068 ms
[root@c2 ~]# exit
exit
vagrant@ubuntu-jammy:~$
```

Ahora, para crear un directorio compartido entre el host y alguno de los contenedores debemos proceder como sigue:

```
mkdir disco1
lxc config device add c2 disco1 disk source=/home/vagrant/disco1 path=/mnt/disco1
```

Y probamos que efectivamente hemos creado un directorio compartido:

```
vagrant@ubuntu-jammy:~$ lxc config device add c2 disco1 disk source=/home/vagrant/disco1 path=/mnt/disco1
Device disco1 added to c2
vagrant@ubuntu-jammy:~$ lxc exec c2 bash
[root@c2 ~]# ls
[root@c2 ~]# cd /mnt/disco1/
[root@c2 disco1]# exit
exit
vagrant@ubuntu-jammy:~$ touch disco1/prueba.txt
vagrant@ubuntu-jammy:~$ lxc exec c2 bash
[root@c2 ~]# cd /mnt/disco1/
[root@c2 disco1]# ls
prueba.txt
[root@c2 disco1]# exit
exit
vagrant@ubuntu-jammy:~$
```

Para dar acceso de lectura y escritura hay que comprobar los permisos del propietario del contenedor mediante `sudo ls -l /var/lib/lxd/containers`.

Para limitar los recursos de un contenedor, en este caso la memoria, y realizar las comprobaciones de la correcta limitación ejecutamos:

```
lxc config set c0 limits.memory 512MB
lxc config show c1
lxc exec c0 bash
[root@c0 ~]# free -m
```

```
exit
vagrant@ubuntu-jammy:~$ lxc config set c0 limits.memory 512MB
vagrant@ubuntu-jammy:~$ lxc config show c0
architecture: x86_64
config:
  image.architecture: amd64
  image.description: ubuntu 24.04 LTS amd64 (release) (20241119)
  image.label: release
  image.os: ubuntu
  image.release: noble
  image.serial: "20241119"
  image.type: squashfs
  image.version: "24.04"
  limits.memory: 512MB
  volatile.base_image: 48d8df642812cf14fceb8db901f30b415d4829e79b82bbccf7dc73f94f0205f6
  volatile.cloud-init.instance-id: a00a8afe8a-0041-4b3f-a100-a43da3b8d0cc
  volatile.eth0.host_name: vethf8dc82fc
  volatile.eth0.hwaddr: 00:16:3e:e1:e3:e7
  volatile.idmap.base: "0"
  volatile.idmap.current: '[{"Isuid":true,"Isgid":false,"Hostid":1000000,"Nsuid":0,"Maprange":1000000000}, {"Isuid":false,"Isgid":true,"Hostid":1000000,"Nsuid":0,"Maprange":1000000000}]'
  volatile.idmap.next: '[{"Isuid":true,"Isgid":false,"Hostid":1000000,"Nsuid":0,"Maprange":1000000000}, {"Isuid":false,"Isgid":true,"Hostid":1000000,"Nsuid":0,"Maprange":1000000000}]'
  volatile.last_state.idmap: '[{"Isuid":true,"Isgid":false,"Hostid":1000000,"Nsuid":0,"Maprange":1000000000}, {"Isuid":false,"Isgid":true,"Hostid":1000000,"Nsuid":0,"Maprange":1000000000}]'
  volatile.last_state.power: RUNNING
  volatile.uuid.generation: 7bd63852-feaa-40e0-b923-ce900d5fc03a
devices: {}
ephemeral: false
profiles:
- default
stateful: false
description: ""
vagrant@ubuntu-jammy:~$ lxc exec c0 bash
root@c0:~# free -m
              total        used        free      shared  buff/cache   available
Mem:           488          36         395           0          55         451
Swap:              0              0              0
root@c0:~# exit
exit
vagrant@ubuntu-jammy:~$
```

Crearemos ahora un *backup* de uno de nuestros contenedores y realizaremos algunos de los comandos que se muestran en el gui3n:

```
lxc snapshot c0 backup_c0
lxc restore c0 backup_c0
lxc move c0/backup_c0 c0/backup-c0
lxc info c0
lxc copy c0/backup-c0 c1
lxc delete c0/backup-c0
```

```
vagrant@ubuntu-jammy:~$ lxc snapshot c0 backup_c0
vagrant@ubuntu-jammy:~$ lxc restore c0 backup_c0
vagrant@ubuntu-jammy:~$ lxc move c0/backup_c0 c0/backup-c0
vagrant@ubuntu-jammy:~$ lxc info c0
Name: c0
Status: RUNNING
Type: container
Architecture: x86_64
PID: 11020
Created: 2024/12/10 09:10 UTC
Last Used: 2024/12/10 10:46 UTC

Resources:
Processes: 24
Disk usage:
  root: 11.11MiB
CPU usage:
  CPU usage (in seconds): 5
Memory usage:
  Memory (current): 93.38MiB
Network usage:
  eth0:
    Type: broadcast
    State: UP
    Host interface: veth05ec463
    MAC address: 00:16:3e:e1:e3:e7
    MTU: 1500
    Bytes received: 1.21kB
    Bytes sent: 1.69kB
    Packets received: 10
    Packets sent: 18
    IP addresses:
      inet: 10.75.48.207/24 (global)
      inet6: fd42:e79:44e5f:b296:216:3eff:fee1:e3e7/64 (global)
      inet6: fe80::216:3eff:fee1:e3e7/64 (link)
  lo:
    Type: loopback
    State: UP
    MTU: 65536
    Bytes received: 2968
    Bytes sent: 2968
    Packets received: 4
    Packets sent: 4
    IP addresses:
      inet: 127.0.0.1/8 (local)
      inet6: ::1/128 (local)

Snapshots:
+-----+-----+-----+-----+
| NAME   | TAKEN AT | EXPIRES AT | STATEFUL |
+-----+-----+-----+-----+
| backup-c0 | 2024/12/10 10:44 UTC | | NO |
+-----+-----+-----+-----+
```



```
vagrant@ubuntu-jammy:~$ lxc copy c0/backup-c0 c1
vagrant@ubuntu-jammy:~$ lxc list
```

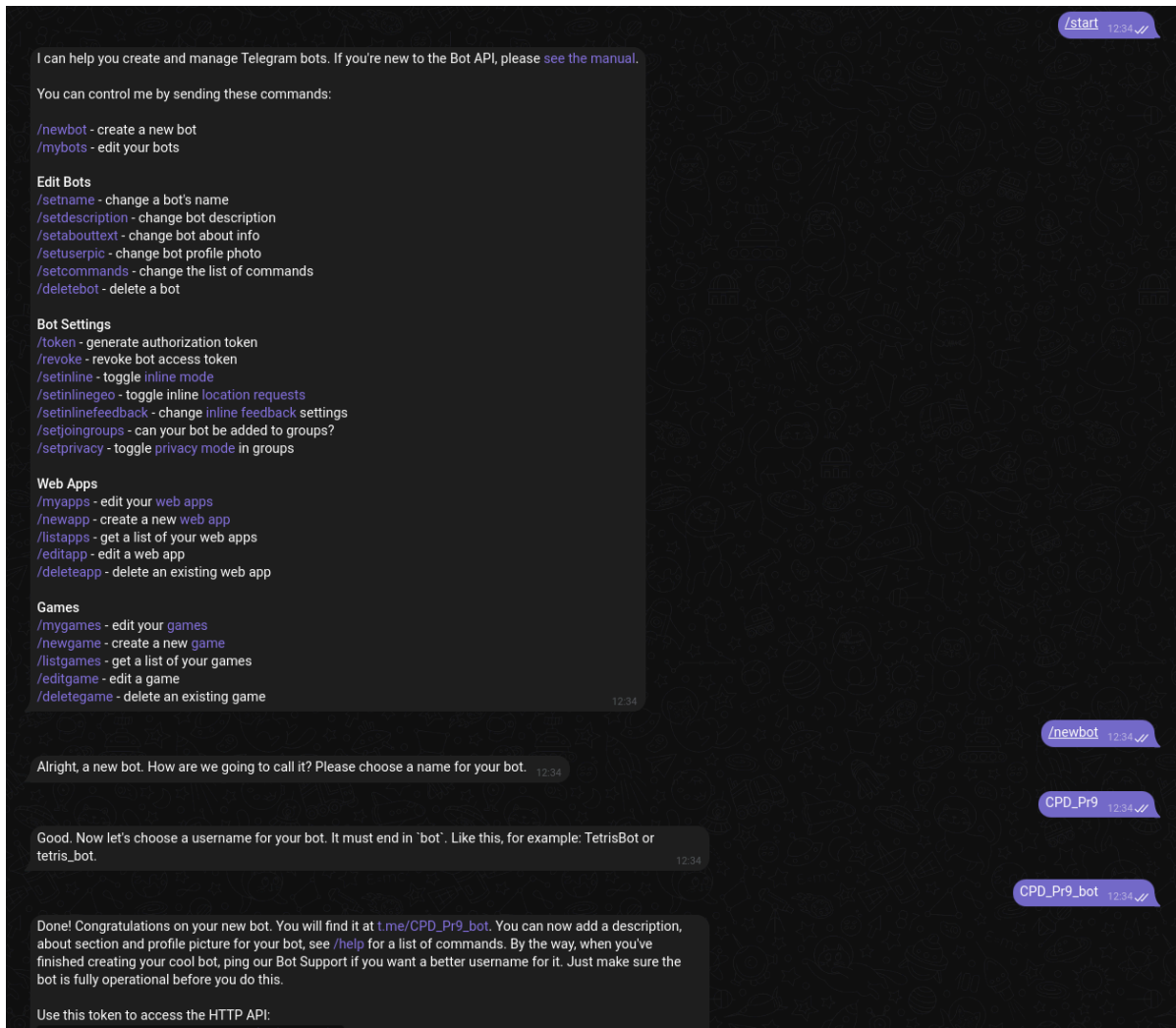
NAME	STATE	IPV4	IPV6	TYPE	SNAPSHOTS
c0	RUNNING	10.75.48.207 (eth0)	fd42:e794:4e5f:b296:216:3eff:fee1:e3e7 (eth0)	CONTAINER	1
c1	STOPPED			CONTAINER	0
c2	RUNNING	10.75.48.118 (eth0)	fd42:e794:4e5f:b296:9638:7d0e:a242:59 (eth0)	CONTAINER	0

```
vagrant@ubuntu-jammy:~$ lxc delete c0/backup-c0
vagrant@ubuntu-jammy:~$
```

Para añadir una interfaz en modo *bridge* a uno de nuestros contenedores ejecutamos: `lxc config device add c2 eth1 nic nictype=bridged parent=lxdbr0`.

Uso de Bot de Telegram

Antes de nada vamos a crear el bot haciendo uso del usuario *BotFather*:



Una vez creado ejecutamos los siguientes comandos para tener todo listo para la creación de cada uno de los 4 bots:

```
cd Bot_Telegram
python3 -m venv env
source env/bin/activate
sudo apt install libssl-dev
pip install python-telegram-bot --pre
pip install command
```

Bot1

El código del primer bot será el dado en el guión:

```
from telegram import Update
from telegram.ext import ApplicationBuilder, CommandHandler, ContextTypes
import command

def ejecuta_w():
    res = command.run(['ls'])
    print(res.output)
    return str(res.output)

async def hello(update: Update, context: ContextTypes.DEFAULT_TYPE) -> None:
    await update.message.reply_text(f'Hello {update.effective_user.first_name}')
async def resp_ls(update: Update, context: ContextTypes.DEFAULT_TYPE) -> None:
    await update.message.reply_text(ejecuta_w())

app =
ApplicationBuilder().token("8012205959:AAHTKUKWwEEfKNnRIjetpLkUHbzFaMMdSI").build()
app.add_handler(CommandHandler("hello", hello))
app.add_handler(CommandHandler("ls", resp_ls))
app.run_polling()
```

Bot2

Para crear el bot que cuente con la función para comprobar la respuesta de ping haremos uso del paquete `icmplib`. También necesitaremos habilitar nuestro proceso pueda usar sockets de red:

```
pip install icmplib
echo 'net.ipv4.ping_group_range = 0 2147483647' | sudo tee -a /etc/sysctl.conf
sudo sysctl -p
```

Una vez tenemos todo esto ya podremos usar el bot2:

```
from telegram import Update
from telegram.ext import ApplicationBuilder, CommandHandler, ContextTypes
from telegram.constants import ParseMode
import command
from icmplib import ping, NameLookupError, ICMPError

def ejecuta_w():
    res = command.run(['ls'])
    print(res.output)
    return str(res.output)

def check_ping(host):
    try:
        return ping(host, privileged=False, count=1).is_alive
    except NameLookupError:
        return "DNS_ERROR"
```



```

except ICMPError:
    return "ICMP_ERROR"
except Exception:
    return "UNKNOWN_ERROR"

async def start(update: Update, context: ContextTypes.DEFAULT_TYPE) -> None:
    welcome_message = (
        "*Este es el Bot de Manuel Díaz-Meco creado para la práctica 9 de la  

asignatura CPD.*\n\n"
        "Estos son los comandos:\n\n"
        "• `/start` - Muestra este mensaje de bienvenida.\n"
        "• `/ping <dirección_ip_o_dominio>` - Realiza un ping al host  

especificado y comprueba si responde.\n\n"
    )
    await update.message.reply_text(text=welcome_message,
    parse_mode=ParseMode.MARKDOWN)

async def resp_ls(update: Update, context: ContextTypes.DEFAULT_TYPE) -> None:
    await update.message.reply_text(ejecuta_w())

async def ping_command(update: Update, context: ContextTypes.DEFAULT_TYPE) ->
None:
    if len(context.args) == 0:
        message = ("Por favor, proporcionar una dirección IP o dominio después  

del comando /ping\n\n"
            "El uso es el siguiente: `/ping <dirección_ip_o_dominio>`")
        await update.message.reply_text(text=message,
        parse_mode=ParseMode.MARKDOWN)
        return

    host = context.args[0]

    result = check_ping(host)

    if result is True:
        await update.message.reply_text(text=f"El host `{host}` está  

respondiendo", parse_mode=ParseMode.MARKDOWN)
    elif result == "DNS_ERROR":
        await update.message.reply_text(text=f"No se pudo resolver el nombre de  

dominio: `{host}`", parse_mode=ParseMode.MARKDOWN)
    elif result == "ICMP_ERROR":
        await update.message.reply_text(text=f"Ocurrió un error ICMP al intentar  

hacer ping a `{host}`", parse_mode=ParseMode.MARKDOWN)
    elif result == "UNKNOWN_ERROR":
        await update.message.reply_text(text=f"Ocurrió un error desconocido al  

intentar hacer ping a {host}", parse_mode=ParseMode.MARKDOWN)
    else:
        await update.message.reply_text(text=f"No se pudo hacer ping al host  

{host}.", parse_mode=ParseMode.MARKDOWN)

app =
ApplicationBuilder().token("8012205959:AAHTKUKWwEEfKNnRIjetpLkUHbzFaMMDDdSI").buil
d()
app.add_handler(CommandHandler("start", start))
app.add_handler(CommandHandler("ping", ping_command))

```

```
app.run_polling()
```

Bot3

Ahora modificaremos el código anterior para que se haga una comprobación con `ping` a la página oficial de la ugr cada minuto. Tendremos que utilizar el paquete `python-telegram-bot[job-queue]`:

```
pip install "python-telegram-bot[job-queue]"
```

Ahora, podemos ejecutar el bot3:

```
from datetime import timedelta
from telegram import Update
from telegram.ext import ApplicationBuilder, CommandHandler, ContextTypes
from telegram.constants import ParseMode
import command
from icmplib import ping, NameLookupError, ICMPError

def ejecuta_w():
    res = command.run(['ls'])
    print(res.output)
    return str(res.output)

def check_ping(host):
    try:
        return ping(host, privileged=False, count=1).is_alive
    except NameLookupError:
        return "DNS_ERROR"
    except ICMPError:
        return "ICMP_ERROR"
    except Exception:
        return "UNKNOWN_ERROR"

async def start(update: Update, context: ContextTypes.DEFAULT_TYPE) -> None:
    welcome_message = (
        "*Este es el Bot de Manuel Díaz-Meco creado para la práctica 9 de la asignatura CPD.*\n\n"
        "Estos son los comandos:\n\n"
        "• `/start` - Muestra este mensaje de bienvenida.\n"
        "• `/ping <dirección_ip_o_dominio>` - Realiza un ping al host especificado y comprueba si responde.\n\n"
        "• `/start_ping_ugr` - Realiza un monitoreo a `ugr.es`. Serás notificado del estado del servidor cada minuto.\n\n"
        "• `/stop_ping_ugr` - Dejarás de estar notificado del estado del servidor `ugr.es` cada minuto.\n\n"
    )
    await update.message.reply_text(text=welcome_message,
    parse_mode=ParseMode.MARKDOWN)
```

```

async def resp_ls(update: Update, context: ContextTypes.DEFAULT_TYPE) -> None:
    await update.message.reply_text(ejecuta_w())

async def ping_command(update: Update, context: ContextTypes.DEFAULT_TYPE) ->
None:
    if len(context.args) == 0:
        message = ("Por favor, proporcionar una dirección IP o dominio después
del comando /ping\n\n"
                    "El uso es el siguiente: `/ping <dirección_ip_o_dominio>`)")
        await update.message.reply_text(text=message,
                                        parse_mode=ParseMode.MARKDOWN)
        return

    host = context.args[0]

    result = check_ping(host)

    if result is True:
        await update.message.reply_text(text=f"El host `{host}` está
respondiendo", parse_mode=ParseMode.MARKDOWN)
    elif result == "DNS_ERROR":
        await update.message.reply_text(text=f"No se pudo resolver el nombre de
dominio: `{host}`", parse_mode=ParseMode.MARKDOWN)
    elif result == "ICMP_ERROR":
        await update.message.reply_text(text=f"Ocurrió un error ICMP al intentar
hacer ping a `{host}`", parse_mode=ParseMode.MARKDOWN)
    elif result == "UNKNOWN_ERROR":
        await update.message.reply_text(text=f"Ocurrió un error desconocido al
intentar hacer ping a {host}", parse_mode=ParseMode.MARKDOWN)
    else:
        await update.message.reply_text(text=f"No se pudo hacer ping al host
{host}.", parse_mode=ParseMode.MARKDOWN)

# Función nueva: Hacer check a la dirección de la ugr
async def check_ugr_ping(context: ContextTypes.DEFAULT_TYPE):
    chat_id = context.job.data['chat_id']
    result = check_ping('ugr.es') #Hacemos ping a ugr.es
    if result is True:
        message = "El host `ugr.es` está respondiendo"
    elif result == "DNS_ERROR":
        message = "No se pudo resolver el nombre de dominio: `ugr.es`"
    elif result == "ICMP_ERROR":
        message = "Ocurrió un error ICMP al intentar hacer ping a `ugr.es`"
    elif result == 'UNKNOWN_ERROR':
        message = "Ocurrió un error desconocido al intentar hacer ping a
`ugr.es`"
    else:
        message = "No se pudo hacer ping al host `ugr.es`"
    await context.bot.send_message(chat_id=chat_id, text=message,
parse_mode=ParseMode.MARKDOWN)

# Función nueva: Hacer ping a la dirección de la ugr cada minuto
async def start_ugr_ping(update: Update, context: ContextTypes.DEFAULT_TYPE):
    chat_id = update.effective_chat.id

```

```

        context.job_queue.run_repeating(check_ugr_ping,
interval=timedelta(minutes=1), first=10, data={'chat_id': chat_id})
        await update.message.reply_text("Se ha iniciado el monitoreo de ping a
`ugr.es`. Recibirás actualizaciones cada minuto.")

# Función nueva: Dejar de hacer ping a la dirección de la ugr
async def stop_ugr_ping(update: Update, context: ContextTypes.DEFAULT_TYPE):
    chat_id = update.effective_chat.id
    current_jobs = context.job_queue.get_jobs_by_name('check_ugr_ping')
    job_removed = False
    for job in current_jobs:
        if job.data['chat_id'] == chat_id:
            job.schedule_removal()
            job_removed = True
            break
    if job_removed:
        await update.message.reply_text("Se ha detenido el monitoreo de ping
a `ugr.es` para este chat", parse_mode= ParseMode.MARKDOWN)
    else:
        await update.message.reply_text("No se encontró ningún monitoreo
activo para este chat")

app =
ApplicationBuilder().token("8012205959:AAHTKUKWwEEfKNnRIjetpLkUHbzFaMMDDsI").buil
d()
app.add_handler(CommandHandler("start", start))
app.add_handler(CommandHandler("ping", ping_command))
app.add_handler(CommandHandler("start_ping_ugr", start_ugr_ping))
app.add_handler(CommandHandler("stop_ugr_ping", stop_ugr_ping))
app.run_polling()

```

Bot4

Esta es la versión final de nuestro bot, podrá ejecutar comandos en la *shell* de nuestro ordenador y mandar la respuesta al bot de Telegram

```

from datetime import timedelta
from telegram import Update
from telegram.ext import ApplicationBuilder, CommandHandler, ContextTypes
from telegram.constants import ParseMode
import command
from icmplib import ping, NameLookupError, ICMPError

# Función nueva: Ejecuta el comando que se indique tras /w
def ejecuta_w(comando):
    try:
        res = command.run([comando])
        print(res.output)
        return str(res.output)
    except Exception as e:
        print(f"Ocurrió un error inesperado: {e}")
        return f"Error inesperado: {e}"

```

```

def check_ping(host):
    try:
        return ping(host, privileged=False, count=1).is_alive
    except NameLookupError:
        return "DNS_ERROR"
    except ICMPError:
        return "ICMP_ERROR"
    except Exception:
        return "UNKNOWN_ERROR"

async def start(update: Update, context: ContextTypes.DEFAULT_TYPE) -> None:
    welcome_message = (
        "*Este es el Bot de Manuel Díaz-Meco creado para la práctica 9 de la  

asignatura CPD.*\n\n"
        "Estos son los comandos:\n\n"
        "• `/start` - Muestra este mensaje de bienvenida.\n"
        "• `/ping <dirección_ip_o_dominio>` - Realiza un ping al host  

especificado y comprueba si responde.\n\n"
        "• `/start_ping_ugr` - Realiza un monitoreo a `ugr.es`. Serás notificado  

del estado del servidor cada minuto.\n\n"
        "• `/stop_ping_ugr` - Dejarás de estar notificado del estado del servidor  

`ugr.es` cada minuto.\n\n"
        "• `/w <ordenes_a_ejecutar>` - Procesará las órdenes dadas por el shell y  

devolverá la respuesta.\n\n"
    )
    await update.message.reply_text(text=welcome_message,
    parse_mode=ParseMode.MARKDOWN)

async def resp_ls(update: Update, context: ContextTypes.DEFAULT_TYPE) -> None:
    await update.message.reply_text(ejecuta_w())

async def ping_command(update: Update, context: ContextTypes.DEFAULT_TYPE) ->
None:
    if len(context.args) == 0:
        message = ("Por favor, proporcionar una dirección IP o dominio después  

del comando /ping\n\n"
        "El uso es el siguiente: `/ping <dirección_ip_o_dominio>`")
        await update.message.reply_text(text=message,
        parse_mode=ParseMode.MARKDOWN)
        return

    host = context.args[0]

    result = check_ping(host)

    if result is True:
        await update.message.reply_text(text=f"El host `{host}` está  

respondiendo", parse_mode=ParseMode.MARKDOWN)
    elif result == "DNS_ERROR":
        await update.message.reply_text(text=f"No se pudo resolver el nombre de  

dominio: `{host}`", parse_mode=ParseMode.MARKDOWN)
    elif result == "ICMP_ERROR":
        await update.message.reply_text(text=f"Ocurrió un error ICMP al intentar  

hacer ping a `{host}`", parse_mode=ParseMode.MARKDOWN)
    elif result == "UNKNOWN_ERROR":

```

```

        await update.message.reply_text(text=f"Ocurrió un error desconocido al
intentar hacer ping a {host}", parse_mode=ParseMode.MARKDOWN)
    else:
        await update.message.reply_text(text=f"No se pudo hacer ping al host
{host}.", parse_mode=ParseMode.MARKDOWN)

async def check_ugr_ping(context: ContextTypes.DEFAULT_TYPE):
    chat_id = context.job.data['chat_id']
    result = check_ping('ugr.es') #Hacemos ping a ugr.es
    if result is True:
        message = "El host `ugr.es` está respondiendo"
    elif result == "DNS_ERROR":
        message = "No se pudo resolver el nombre de dominio: `ugr.es`"
    elif result == "ICMP_ERROR":
        message = "Ocurrió un error ICMP al intentar hacer ping a `ugr.es`"
    elif result == 'UNKNOWN_ERROR':
        message = "Ocurrió un error desconocido al intentar hacer ping a
`ugr.es`"
    else:
        message = "No se pudo hacer ping al host `ugr.es`"
    await context.bot.send_message(chat_id=chat_id, text=message,
parse_mode=ParseMode.MARKDOWN)

async def start_ugr_ping(update: Update, context: ContextTypes.DEFAULT_TYPE):
    chat_id = update.effective_chat.id
    context.job_queue.run_repeating(check_ugr_ping,
interval=timedelta(minutes=1), first=10, data={'chat_id': chat_id})
    await update.message.reply_text("Se ha iniciado el monitoreo de ping a
`ugr.es`. Recibirás actualizaciones cada minuto.")

async def stop_ugr_ping(update: Update, context: ContextTypes.DEFAULT_TYPE):
    chat_id = update.effective_chat.id
    current_jobs = context.job_queue.get_jobs_by_name('check_ugr_ping')
    job_removed = False
    for job in current_jobs:
        if job.data['chat_id'] == chat_id:
            job.schedule_removal()
            job_removed = True
            break
    if job_removed:
        await update.message.reply_text("Se ha detenido el monitoreo de ping a
`ugr.es` para este chat", parse_mode= ParseMode.MARKDOWN)
    else:
        await update.message.reply_text("No se encontró ningún monitoreo activo
para este chat")

# Función nueva: Permite correr un comando de la shell de nuestro ordenador

async def correr_orden(update: Update, context: ContextTypes.DEFAULT_TYPE) ->
None:
    if (context.args) == 0:
        message = ("Por favor, proporcionar una orden después del comando /w\n\n")

```



```

        "El uso es el siguiente: `/w <ordenes_a_ejecutar>`")
        await
    update.message.reply_text(text=message, parse_mode=ParseMode.MARKDOWN)
    return
    orden = context.args[0]
    await update.message.reply_text(ejecuta_w(orden))

app =
ApplicationBuilder().token("8012205959:AAHTKUKWwEEfKNnRIjetpLkUHbzFaMMDDsI").build()
app.add_handler(CommandHandler("start", start))
app.add_handler(CommandHandler("ping", ping_command))
app.add_handler(CommandHandler("start_ping_ugr", start_ugr_ping))
app.add_handler(CommandHandler("stop_ping_ugr", stop_ugr_ping))
app.add_handler(CommandHandler("w", correr_orden))
app.run_polling()

```

