4. Storage, Backup y BBDD

Para estos laboratorios se han preparado dos entornos en los directorios vagrant_disk y vagrant_lvm. El primero consiste en dos máquinas virtuales, una sobre CentOS y otra sobre Ubuntu. En ambas máquinas se va a crear un disco adicional de 1GB. En el segundo entorno vamos a montar una máquina con ubuntu donde se añadirán hasta 4 discos adicionales de 1GB.

1. Montar una ISO en la máquina virtual

a. Conectarse por ssh a la máquina con **ubuntu**
b. Instalar el paquete genisoimage con **sudo apt install genisoimage**
c. Creamos una imagen ISO del directorio /vagrant ejecutando **mkisofs -lJR -o /vagrant/vagrant.iso /vagrant**.
d. Hemos guardado la imagen en el directorio /vagrant con lo que desde nuestra máquina podemos verificarla. Si no también la podemos verificar desde la línea de comandos:

```
vagrant@ubuntu:~$ file /vagrant/vagrant.iso
/vagrant/vagrant.iso: ISO 9660 CD-ROM filesystem data 'CDROM'
```

e. Creamos un nuevo directorio, /vagrant2 y montamos la imagen ejecutando: **sudo mount -o loop -t iso9660 /vagrant/vagrant.iso /vagrant2**
f. Podremos verificar que se nos ha montado la iso con el comando **df -T | grep vagrant**
g. Ambos puntos de montaje deberían tener el mismo contenido (a excepción de la imagen iso). Para ello ejecutamos tree /vagrant y tree /vagrant2 y debería ser una salida idéntica.
h. Para desmontar la imagen iso ejecutamos **sudo umount /vagrant2**.

2. Verificar tablas de particiones MBR y GPT

a. Arrancar la VM con **CentOS** y conectarnos a ella como root.
b. Si ejecutamos un **fdisk -l** podremos ver los discos que ve la máquina virtual. Tendremos /dev/sda con el “Disk Label Type: dos” lo que indica que usa una partición MBR. Y además veremos que existe la partición /dev/sda1.
c. El disco /dev/sdb no está inicializado con lo que no tiene ninguna tabla de particiones.
d. Vamos a particionarlo ejecutando **fdisk /dev/sdb**. Veremos que nos muestra un mensaje de que va a crear una tabla de particiones tipo DOS. Para cambiarlo pulsamos **g** y cambiará a GPT. La salida nos devolverá el GUID del disco.
e. Guardaremos los cambios con **w** y ejecutaremos de nuevo **fdisk -l** para verificar si ha cambiado.
f. ¿Qué pasaría si hiciéramos el mismo cambio en /dev/sda?

3. Añadir nuevo filesystem y configurarlo para que se monte al arrancar

a. Conectarse por ssh a la máquina con **ubuntu** y pasarse a root
b. Con el **fdisk -l** debemos ver que tenemos un disco /dev/sdc que no está particionado.
c. Ejecutamos **fdisk /dev/sdc** para particionarlo.
d. Pulsaremos **n** para crear una nueva partición. Nos pedirá si queremos que sea primaria o extendida. En este caso pulsaremos **p** para que cree una primaria. Nos pedirá ahora que le indiquemos un número de partición, al ser la primera le diremos que **1**. A continuación debemos dimensionar la partición. Ya que nos interesa utilizar todo el disco pulsaremos intro para aceptar los valores por defecto de primer sector y último sector.
e. Si pulsamos **p** podremos verificar la tabla de particiones que hemos definido. Por defecto el tipo de partición que nos ha añadido es Linux con lo que nos servirá para continuar.
f. El último paso con fdisk es guardar los cambios pulsando **w**.
g. Con **ls -l /dev/sdc*** veremos que ya tenemos un nuevo dispositivo, en este caso el sdc1 que corresponde a la partición que acabamos de crear.
h. El siguiente paso será formatearla con algún tipo de sistema de fichero para poder montarla. La formatearemos con ext4 con el comando **mkfs.ext4 /dev/sdc1**.
i. Una vez tenemos la partición formateada crearemos el directorio /keepcoding (**mkdir /kepcoding**) y la montaremos de la siguiente manera **mount /dev/sdc1 /keepcoding**. Con **df -T /keepcoding** podremos verificar si se ha montado correctamente.
j. Para hacer el cambio persistente deberemos añadir la entrada correspondiente en el fichero /etc/fstab. Para eso editamos el fichero /etc/fstab y añadimos la siguiente línea

```
/dev/sdc1       /keepcoding     ext4    defaults        0 0
```

k. A continuación reiniciamos la máquina y verificamos si se ha montado el filesystem en el directorio /keepcoding
l. Extra: en lugar de indicar /dev/sdc1 en el fichero /etc/fstab indicar el UUID (se obtiene con blkid)

4. Añadir nueva partición de swap

a. Para comenzar con la máquina en el estado inicial haremos un **vagrant destroy ubuntu** y un **vagrant up ubuntu**.
b. Ejecutamos los mismos pasos que en el Ejercicio 4.3 hasta el punto d. En este caso debemos cambiar el tipo de partición. Para ello pulsamos **t** justo después de crear la partición y le indicamos el tipo **82** correspondiente a Linux Swap. Podemos ver un listado de los tipos de particiones con **L**.

c. Para verificar la partición que hemos creado pulsamos **p** y a continuación grabamos los cambios con **w**.
d. De forma similar al ejercicio anterior, a la partición de swap deberemos formatearla como swap, para ello ejecutamos **mkswap /dev/sdc1**.
e. Con el comando **free** podemos ver que nuestra máquina no tiene ningún espacio de swap asignado. Para activar la partición de swap nueva ejecutamos **swapon /dev/sdc1**. Con **free** podremos verificar que ahora ya tenemos más swap disponible
f. Para hacer estos cambios permanentes deberemos añadirlos en el /etc/fstab añadiendo la siguiente línea:

```
/dev/sdc1      swap    swap    defaults        0 0
```

g. Si reiniciamos la máquina podremos verificar que con el comando **free** que se ha activado la partición de swap (también se puede verificar con el comando **swapon --show**)
h. Extra: en lugar de indicar /dev/sdc1 en el fichero /etc/fstab indicar el UUID (se obtiene con blkid)

5. Extender volumen LVM

a. Para este ejercicio cambiamos al directorio **vagrant_lvm** y arrancamos la máquina con **vagrant up**.
b. Conectarse por ssh a la máquina con ubuntu y pasarse a root
c. Con **ls -l /dev/sd*** veremos que en esta máquina tenemos hasta 6 discos, de los cuales tenemos 4 disponibles.
d. Particionaremos el primero de ellos con **fdisk /dev/sdc** indicando que el tipo de particionado será GPT, con una partición que ocupe todo el disco y del tipo “Linux LVM”. Deberíamos obtener un particionado similar a este:

```
Disk /dev/sdc: 1 GiB, 1073741824 bytes, 2097152 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: gpt
Disk identifier: 28EF8B81-E17E-4095-9EF3-ED12E5AEDE40

Device     Start     End Sectors  Size Type
/dev/sdc1   2048 2097118 2095071 1023M Linux LVM
```

e. Guardamos los cambios con **w**.
f. Para replicar el particionado en el resto de discos (se puede hacer ya que son idénticos) haremos una copia de la tabla de particiones del disco sdc y la restauraremos en el resto. Para hacer la copia ejecutamos **sfdisk -d /dev/sdc > lvm_part_table**
g. Restauraremos la tabla de particiones en el resto de discos:

```
sfdisk /dev/sdd < lvm_part_table
sfdisk /dev/sde < lvm_part_table
sfdisk /dev/sdf < lvm_part_table
```

h. Con **ls -l /dev/sd*1** podremos ver que se nos han creado todas las particiones correctamente.
i. Pasamos a crear el volúmen físico con **pvcreate /dev/sdc1**. Con **pvdisplay** o **pvs** podemos ver la lista de volúmenes físicos configurados.
j. A continuación creamos un Volume Group nuevo al que le añadimos este volúmen físico recién creado. Para ello ejecutamos: **vgcreate keepcoding_vg /dev/sdc1**. Con **vgdisplay** o **vgs** demos los Volume Group configurados.
k. Una vez tenemos el Volume Group podemos crear un volumen lógico. En este caso crearemos uno con todo el tamaño disponible de la siguiente manera: **lvcreate -l 100%FREE keepcoding_vg -n keepcoding_data**. Podemos verificar que nos ha creado el volumen con **lvdisplay** o **lvs**. También podemos ver como nos ha creado un nuevo dispositivo en **/dev/keepcoding_vg/keepcoding_data**.
l. Como en el Ejercicio 4.3, debemos formatear el dispositivo para poder montarlo. Para ello utilizamos de nuevo el comando mkfs.ext4 esta vez contra el volumen lógico: **mkfs.ext4 /dev/keepcoding_vg/keepcoding_data**.
m. Ya tendremos el dispositivo listo para su montaje con **mount /dev/keepcoding_vg/keepcoding_data /keepcoding**. Podremos ver con un df -h que se ha montado y tenemos aproximadamente 1GB disponible.
n. A continuación vamos a extender este volumen para tener más espacio disponible en el filesystem. Para ello crearemos el volúmen físico con pvcreate del dispositivo /dev/sdd y lo añadiremos al Volume Group keepcoding_vg ejecutando: **vgextend keepcoding_vg /dev/sdd1**. En la salida del **vgdisplay** veremos que el parámetro VG Size se ha incrementado.
o. Ahora asignaremos el espacio que tenemos disponible en el Volume Group al volúmen lógico con **lvextend -l +100%FREE /dev/keepcoding_vg/keepcoding_data**. El volúmen lógico habrá pasado de tener 1G a 2G aproximadamente. El último paso será extender el filesystem con **resize2fs /dev/keepcoding_vg/keepcoding_data**. Este cambio lo hemos podido realizar en caliente y sin pérdida de datos.
p. EXTRA: configurar para que se monte automáticamente el filesystem al arrancar el sistema.
q. EXTRA 2: Para reducir el tamaño de un sistema de ficheros no lo podemos hacer en caliente y deberíamos desmontar el filesystem. Para reducirlo en el ejemplo anterior:

```
root@ubuntu:~# umount /keepcoding
root@ubuntu:~# lvresize --resizefs --size 800MB
/dev/mapper/keepcoding_vg-keepcoding_data
fsck from util-linux 2.27.1
/dev/mapper/keepcoding_vg-keepcoding_data: 11/130560 files (0.0%
non-contiguous), 12684/522240 blocks
resize2fs 1.42.13 (17-May-2015)
Resizing the filesystem on
/dev/mapper/keepcoding_vg-keepcoding_data to 204800 (4k) blocks.
```

The filesystem on /dev/mapper/keepcoding_vg-keepcoding_data is now 204800 (4k) blocks long.

Size of logical volume keepcoding_vg/keepcoding_data changed from 1.99 GiB (510 extents) to 800.00 MiB (200 extents).
Logical volume keepcoding_data successfully resized.

6. Configurar un mirror con LVM de dos discos

a. Para este ejercicio cambiamos al directorio **vagrant_lvm** y arrancamos la máquina con **vagrant up**.
b. Conectarse por ssh a la máquina con ubuntu y pasarse a root.
c. Eliminamos el volúmen lógico keepcoding_data para tener espacio disponible en el Volume Group. Para ello ejecutamos **lvremove /dev/keepcoding_vg/keepcoding_data**. Nota: Podríamos haber reducido el volumen para tener algo de espacio disponible.
d. En este punto tenemos un Volume Group formado por dos volúmenes físicos sdc1 y sdd1. A continuación crearemos un nuevo volumen de 100MB donde le indicaremos que debe ser un mirrror, con lo que escribirá el mismo dato simultáneamente en los dos discos: **lvcreate --mirrors 1 --type raid1 -L 100MB -n keepcoding_mirror keepcoding_vg**.
e. Con **vgdisplay** podemos ver que el espacio libre es de 1800MB aproximadamente y con el lvdisplay podremos ver las características de este volumen.
f. Como en el punto anterior procederemos a crear el sistema de ficheros y montarlo. Con **df** verificamos que tiene las características que queríamos.
g. Si por ejemplo quisiéramos liberar un volúmen físico, deberíamos añadir otro antes al volume group, ya que por las características del volumen no nos lo permite:

```
root@ubuntu:~# pvmove /dev/sdd1
  Insufficient free space: 26 extents needed, but only 0 available
  Unable to allocate mirror extents for pvmove0.
  Failed to convert pvmove LV to mirrored
```

h. Para añadir un nuevo volumen físico lanzaríamos el comando **vgextend keepcoding_vg /dev/sde1** y a continuación si que podríamos ejecutar el **pvmove** para liberar el disco sdd1.
i. Para sacar el disco sdd1 del volume group ejecutaríamos **vgreduce keepcoding_vg /dev/sdd1**. De esta forma ya podríamos eliminar el disco sdd1 del sistema.

7. Configurar un stripe con LVM de tres discos

a. En el Ejercicio 4.5 hemos creado un volumen lógico que luego hemos extendido añadiendo un volumen físico. En este ejercicio añadiremos un tercer volumen físico pero a diferencia de lo ejecutado en el Ejercicio 4.5, crearemos el volumen indicando que tiene que repartir los datos equitativamente entre los 3 discos, de esta forma aumenta el performance de las escrituras.
b. Nos conectamos a la máquina con ubuntu y nos aseguramos que tenemos por lo menos los volúmenes físicos sdc1 y sdd1. Añadiremos un tercer volumen físico con **vgextend keepcoding_vg /dev/sde1**. De forma que a la salida del **vgdisplay -v** tengamos algo similar a esto:

```
--- Physical volumes ---
PV Name               /dev/sdc1
PV UUID               VLZ3n3-fF2G-BOBD-BypD-GxG0-fepn-5CWMmc
PV Status             allocatable
Total PE / Free PE    255 / 220

PV Name               /dev/sde1
PV UUID               3rdoYX-WjWb-5f0L-2j2a-4YB0-VcOn-tI9i9I
PV Status             allocatable
Total PE / Free PE    255 / 220

PV Name               /dev/sdd1
PV UUID               q8qSbi-DLRV-BNWp-mrJ4-fMtb-2nzN-NZ4D1v
PV Status             allocatable
Total PE / Free PE    255 / 246
```

c. Creamos un volumen indicando que será de tipo stripe: **lvcreate --stripes 3 -L 100MB -n keepcoding_stripe keepcoding_vg**.
d. Formateamos el volumen y lo montamos en /keepcoding para verificar que se ha creado correctamente.
e. En la salida del **vgdisplay -v** veremos que el nuevo volumen tiene asignados 27 extents (Current LE). Este número será múltiplo del número de stripes definidos al crear el volumen. En este caso asigna 9 extents a cada volumen físico.

8. Configurar un RAID5 con LVM.

a. En este caso deberemos añadir un cuarto volumen físico con el comando **vgextend**.
b. Una vez añadido creamos el volumen lógico en raid5 de la siguiente manera: **lvcreate --type raid5 -L 100MB -n keepcoding_raid5 keepcoding_vg**.
c. Si hemos eliminado los volúmenes lógicos que hemos creado en los ejercicios anteriores podemos ver claramente cómo han quedado los extents repartidos, 9 en cada dispositivo físico. En cambio cuando listamos las

propiedades del volumen lógico nos indica que en uso tiene 27, ya que no muestra los de paridad.

9. Realizar backups de ficheros con diferentes herramientas

a. Nos conectamos como root a la máquina sobre la que hemos estado creando los volúmenes lógicos en los ejercicios anteriores y montaremos los volúmenes de la siguiente forma:

```
root@ubuntu:~# df -T | grep keepcoding
/dev/mapper/keepcoding_vg-keepcoding_mirror ext4        95054
1550     86336   2% /restore
/dev/mapper/keepcoding_vg-keepcoding_stripe ext4       102997
1550     93707   2% /keepcoding
```

b. Nos moveremos al directorio /keepcoding y haremos en él un backup del directorio /etc, uno de los más importantes del sistema, mediante la herramienta tar: **tar -cf etc_backup.tar /etc**.

c. Para verificar el contenido del archivo podemos listar los ficheros que contiene: **tar tf etc_backup.tar | head**

d. Al tar que hemos creado le añadiremos una copia de los ficheros del directorio /vagrant, a excepción de los ficheros vmdk que pudiera haber: **tar -rvf etc_backup.tar /vagrant --exclude "*.vmdk"**

e. Con **tar -xf etc_backup.tar -C /restore** extraemos la copia en el directorio /restore (crearemos antes el directorio /restore)

f. Volveremos al directorio /keepcoding y extraeremos únicamente el fichero etc/hosts. Verificamos con **tar -tf etc_backup.tar | grep etc/hosts** que existe el fichero en el archivo y lo extraemos con **tar -xvf etc_backup.tar etc/hosts**

g. Cuando hemos hecho el backup del directorio /vagrant hemos visto que se han copiado algunos ficheros de una carpeta oculta que no nos son útiles y procederemos a borrarlos del archivo con **tar --delete --file etc_backup.tar vagrant/.vagrant**. Volveremos a listar el contenido del archivo para verificar si hay algún fichero que haga referencia a esa carpeta: **tar -tf etc_backup.tar | grep vagrant**

h. A continuación vamos a realizar un backup con la herramienta cpio. Para realizar el backup del directorio /etc concatenamos la salida del comando find de la siguiente forma: **find /etc | cpio -ov > etc.cpio**

i. Podemos verificar el contenido del fichero con **cpio -it < etc.cpio**.

j. Para extraer usaríamos el comando **cpio -id <etc.cpio**. OJO: si en el archivo tenemos la ruta absoluta al extraer reemplaza el contenido original sin confirmación.

k. EXTRA: comprimir directamente al utilizar las herramientas tar y cpio
   i. find /etc | cpio -ov | gzip -c > etc.cpio.gz
   ii. tar - zcvf etc_backup.tar

l. EXTRA 2: hacer backup de una máquina a otra con rsync (utilizar el entorno vagrant de las prácticas de red) rsync -av /etc vagrant@server2:/restore

10. Instalar una BBDD MariaDB

a. Volver al entorno de trabajo vagrant_cli y arrancar la máquina virtual con ubuntu.
b. Nos conectamos a la máquina virtual y cambiamos al usuario root
c. Instalamos mariaDB con el comando **apt install mariadb-server**.
d. A pesar de ser un fork de MySQL aún continúa manteniendo los nombres de servicios, scripts y ficheros de configuración. Por tanto, para ver el estado del servicio ejecutaremos **systemctl status mysql.service**.
e. Una vez instalado correremos el script **mysql_secure_installation** para aplicar unas configuraciones básicas de seguridad: establecer la contraseña de root, borrar usuarios anónimos, deshabilitar el login remoto del usuario root, eliminar la base de datos de test y recargar los privilegios.
f. A continuación ya nos podremos conectar al mysql con **mysql -uroot -p**.
g. Crearemos una base de datos ya con la sintaxis de SQL: **CREATE DATABASE bonus;** Salimos de la shell de MariaDB.
h. Descargamos el siguiente backup: curl -o ms.sql https://raw.githubusercontent.com/Gopalpothuraju/learning-mysql/master/ms.sql
i. Restauramos el backup de esta BBDD en la BBDD bonus que acabamos de crear: **mysql -uroot -p bonus < ms.sql**
j. Podemos volver a la shell de MariaDB y verificar si se ha restaurado correctamente.

11. Instalar una BBDD MongoDB

a. Nos conectamos a la máquina virtual y cambiamos al usuario root
b. La versión de MongoDB que hay en los repositorios es muy antigua, por tanto activaremos los repositorios oficiales. Deberemos añadir la clave del repositorio

    sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv 9DA31620334BD75D9DCB49F368818C72E52529D4

c. A continuación añadimos el repositorio a nuestros orígenes:

    echo "deb https://repo.mongodb.org/apt/ubuntu bionic/mongodb-org/4.0 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-4.0.list

d. Deberemos actualizar la cache de APT con **apt-get update** y instalamos MongoDB con el comando **apt install mongodb-org**.
e. Una vez instalado verificaremos que el servicio ha arrancado: **systemctl status mongod**. En caso de que no lo haya hecho lo arrancamos: **systemctl start mongod** y habilitamos: **systemctl enable mongod**.

f. En los mensajes de log podremos verificar si ha arrancado correctamente o no: **tail -10 /var/log/mongodb/mongod.log**.
g. Finalmente con el comando **mongo** nos podremos conectar y lanzar consultas:
    i. show dbs;
    ii. use demoDB;
    iii. db.demoCol.insertOne({x:1});
    iv. db.demoCol.insertOne({x:1, y:3});
    v. db.demoCol.find()

12. Crear un replicaset en HA con MongoDB

a. Crearemos un Replica Set en una misma VM lanzando tres procesos con un fichero de configuración diferente. Los pasos serán los siguientes.
b. Crear un directorio nuevo con **sudo mkdir /etc/mongodb** y crear un nuevo fichero de configuración /etc/mongodb/nodo1.conf (revisar permisos). En bindIP debemos poner la IP que tiene nuestra VM.

```
storage:
  dbPath: /var/mongodb/db/node1
net:
  bindIp: 10.0.2.15,localhost
  port: 27011
security:
  authorization: enabled
  keyFile: /var/mongodb/pki/kpcd-keyfile
systemLog:
  destination: file
  path: /var/mongodb/db/node1/mongod.log
  logAppend: true
processManagement:
  fork: true
replication:
  replSetName: rs-keepcoding
```

c. Crear la clave usada en la replicación

```
sudo mkdir -p /var/mongodb/pki/
sudo chown -R vagrant:vagrant /var/mongodb/
openssl rand -base64 741 > /var/mongodb/pki/kpcd-keyfile
chmod 400 /var/mongodb/pki/kpcd-keyfile
```

d. Crear el DBpath para el nodo1

```
mkdir -p /var/mongodb/db/node1
```

e. Arrancar un mongo con la configuración del nodo1

```
mongod -f node1.conf
```

f. Copiar la configuración del nodo 1 y replicarla en nodo2.conf y nodo3.conf

```
cp node1.conf node2.conf
cp node2.conf node3.conf
```

g. Editarlas cambiando el dbpath, port, logpath

```
vi node2.conf
vi node3.conf
```

h. Crear el DBpath para los otros nodos

```
mkdir -p /var/mongodb/db/node2
mkdir -p /var/mongodb/db/node3
```

i. Arrancar un proceso de mongo para cada nodo

```
mongod -f node2.conf
mongod -f node3.conf
```

de forma que queden tres procesos mongo arrancados

```
vagrant   2943  1.5 13.0 1132096 64988 ?        SLl  20:03   0:01 mongod -f /etc/node1.conf
vagrant   3027  7.8 12.5 1131088 62856 ?        SLl  20:04   0:00 mongod -f /etc/node2.conf
vagrant   3060 14.2 12.7 1131088 63584 ?        SLl  20:04   0:00 mongod -f /etc/node3.conf
vagrant   3094  0.0  0.1  12916   936 pts/0    S+   20:04   0:00 grep --color=auto mongo
```

j. Conectarse al nodo1 e iniciar el replica set

```
mongo --port 27011
rs.initiate()
```

k. Crear el usuario admin

```
use admin
db.createUser({
  user: "kpcd-admin",
  pwd: "kpcd-pass",
  roles: [
    {role: "root", db: "admin"}
  ]
})
```

l. Desconectarse del mongo y conectarse como admin

```
exit
mongo --host "rs-keepcoding/10.0.2.15:27011" -u "kpcd-admin" \
-p "kpcd-pass" --authenticationDatabase "admin"
```

m. Verificar el estado del replica set

```
rs.status()
```

n. Añadir los otros dos nodos

```
rs.add("10.0.2.15:27012")
rs.add("10.0.2.15:27013")
```

o. Verificación

```
rs.status()
rs.isMaster()
db.serverStatus()['repl']
rs.printReplicationInfo()
```

p. Hacer un failover a otro nodo

```
rs.stepDown()
```