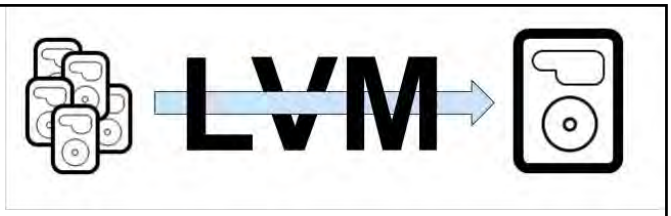


Gestión de LVM en GNU/Linux

Alfredo Abad

ISOP509_LVM.pptx

UA: 8-ene-2023

1

Objetivo de la práctica

- Aprender a gestionar sistemas de información en GNU/Linux con tecnología LVM (Logical Volume Manager)
- Aprender a utilizar ficheros como virtualización de discos físicos mediante las opciones de montado “loop”
- Material necesario
  - Un sistema Ubuntu Server o similar
  - Información del man de Linux

2

Instalación del software necesario

- El núcleo debe incluir soporte para LVM (las nuevas distribuciones ya lo incluyen, versión de kernel 2.6 o superior) y debe disponerse del conjunto de herramientas de gestión de volúmenes en LVM (paquetes lvm1 o, modernamente y preferible, lvm2)
  - sudo **apt-get install lvm2**

3

Preparación de los HD virtuales (se podría hacer también con físicos)

- Creación de los ficheros-disco con **dd**
  - **dd if=/dev/zero of=fichero bs=1M count=100**
    - (se inician con ceros: fichero1 y fichero2)

```
root@ubsrv64:~#
root@ubsrv64:~# dd if=/dev/zero of=fichero0 bs=1M count=100
100+0 registros leídos
100+0 registros escritos
104857600 bytes (105 MB) copiados, 0,45082 s, 233 MB/s
root@ubsrv64:~# dd if=/dev/zero of=fichero1 bs=1M count=100
100+0 registros leídos
100+0 registros escritos
104857600 bytes (105 MB) copiados, 0,726857 s, 144 MB/s
root@ubsrv64:~#
root@ubsrv64:~# ls -ls
total 204804
102404 -rw-r--r-- 1 root root 104857600 abr 13 17:06 fichero0
102400 -rw-r--r-- 1 root root 104857600 abr 13 17:06 fichero1
root@ubsrv64:~#
```

4

Nota: creación de ficheros y truncado

- Creación:
  - Orden dd:
    - dd if=/dev/zero of=testfile.img bs=2G count=1
  - Orden fallocate:
    - fallocate -l 1G testfile1.img
- Truncado de ficheros:
  - Orden truncate (si no existe el fichero, lo crea):
    - truncate -s 2G testfile.img

- Ejemplos en la diapo siguiente
- Más información:
  - https://linuxhandbook.com/create-large-files/

5

```
vagrant@LHB:~$ dd if=/dev/zero of=testfile.img bs=2G count=1
0+1 records in
0+1 records out
2147479552 bytes (2.1 GB, 2.0 GiB) copied, 20.9786 s, 102 MB/s
vagrant@LHB:~$ ls -l testfile.img
-rw-rw-r-- 1 vagrant vagrant 2147479552 Nov 24 13:32 testfile.img
vagrant@LHB:~$
```

1 2

```
vagrant@LHB:~$ truncate -s 2G testfile.img
vagrant@LHB:~$ ls testfile.img
testfile.img
vagrant@LHB:~$ ls -lh testfile.img
-rw-rw-r-- 1 vagrant vagrant 2.0G Nov 24 13:40 testfile.img
vagrant@LHB:~$
```

1 2

```
vagrant@LHB:~$ fallocate -l 1G testfile1.img
vagrant@LHB:~$ ls -lh testfile1.img
-rw-rw-r-- 1 vagrant vagrant 1.0G Nov 25 10:39 testfile1.img
vagrant@LHB:~$
```

6

Asociamos los ficheros virtuales a dispositivos en /dev con losetup

- **losetup** es una orden que permite la gestión de dispositivos de tipo “loop”
  - Los asociamos a /dev/loop0 y /dev/loop1 respectivamente

```
root@ubsrv64:~#
root@ubsrv64:~# losetup  /dev/loop0  fichero0
root@ubsrv64:~#
root@ubsrv64:~# losetup  /dev/loop1  fichero1
root@ubsrv64:~#
root@ubsrv64:~#
```

7

Inicio de los volúmenes

- Para poder utilizar los discos como un volumen físico (PV) deben iniciarse con **pvcreate**

```
root@ubsrv64:~#
root@ubsrv64:~# pvcreate  /dev/loop0
  Physical volume "/dev/loop0" successfully created
root@ubsrv64:~#
root@ubsrv64:~# pvcreate  /dev/loop1
  Physical volume "/dev/loop1" successfully created
root@ubsrv64:~#
root@ubsrv64:~#
```

8

Creación de un VG con los dos PV

- Un VG es un grupo de volúmenes físicos o lógicos que se agrupan para formar un conjunto
- Creamos un VG con **vgcreate**
  - **vgcreate** mi_vg /dev/loop0 /dev/loop1

```
root@ubsrv64:~#
root@ubsrv64:~# vgcreate  mi_vg  /dev/loop0  /dev/loop1
  Volume group "mi_vg" successfully created
root@ubsrv64:~#
root@ubsrv64:~# _
```

9

Activamos el VG con vgchange

- La orden **vgchange** cambia los atributos de un VG
  - Solo cuando un VG está “activo” se puede acceder a los volúmenes lógicos que contenga
  - Lo activamos con vgchange **–a y** mi_vg
  - Se desactivaría con vgchange **–a n** mi_vg
    - Una vez desactivado se puede eliminar el VG con **vgremove** mi_vg

```
root@ubsrv64:~# vgchange  -a  y  mi_vg
  0 logical volume(s) in volume group "mi_vg" now active
root@ubsrv64:~#
```

10

Otras órdenes para VG de LVM

- Añadir volúmenes físicos (PV) a un grupo de volúmenes (VG)
  - **vgextend** mi_vg /dev/loop2
- Visualizar un PV
  - **pvdisplay** /dev/loop2
- Eliminar un volumen físico (PV) de un grupo de volúmenes (VG)
  - Se visualiza su contenido con **pvdisplay** para comprobar que nadie lo usa
  - Si se está usando, se puede utilizar **pvmove** para migrar sus datos a otro PV
  - Se elimina una vez vacío con **vgreduce**

11

Ejemplo de pvdisplay

```
root@ubsrv64:~# pvdisplay  /dev/loop0
  --- Physical volume ---
  PV Name               /dev/loop0
  VG Name               mi_vg
  PV Size               100,00 MiB / not usable 4,00 MiB
  Allocatable           yes
  PE Size               4,00 MiB
  Total PE              24
  Free PE               24
  Allocated PE          0
  PV UUID               qzcedC-R8Ue-NL5J-9EK9-HjXp-Q0bg-7AScvu

root@ubsrv64:~#
root@ubsrv64:~# pvdisplay  /dev/loop1
  --- Physical volume ---
  PV Name               /dev/loop1
  VG Name               mi_vg
  PV Size               100,00 MiB / not usable 4,00 MiB
  Allocatable           yes
  PE Size               4,00 MiB
  Total PE              24
  Free PE               24
  Allocated PE          0
  PV UUID               2Dajnm-tFu0-H2Hx-Pqv4-MZVF-60C1-Q8W4XL

root@ubsrv64:~#
```

12

Creación de un volumen lógico (LV) con lvcreate

- Nota: si **lvcreate** genera errores de kernel puede ser porque le falte el módulo **dm-mod** (device mapper)
  - Se podría cargar con **modprobe dm-mod**
- El tamaño de los LV se define directamente o en unidades de "extends" (que por defecto son 4M, aunque se puede cambiar)
  - Con opción **-L** : -L 10M (10M)
  - Con opción **-l** (ele minúscula, en extends): -l 3 (3x4=12M)

13

Creamos dos VG concretos (-n name -nombre del VL-)

- lvcreate -L 10M **-n primer_vl** mi_vg
- lvcreate -l 3 **-n segundo_vl** mi_vg

```
root@ubsrv64:~#
root@ubsrv64:~# lvcreate  -L 10M  -n primer_lv  mi_vg
  Rounding up size to full physical extent 12,00 MiB
  Logical volume "primer_lv" created
root@ubsrv64:~#
root@ubsrv64:~# lvcreate  -l 3  -n segundo_lv  mi_vg
  Logical volume "segundo_lv" created
root@ubsrv64:~#
root@ubsrv64:~#
```

14

Crear VL de bandas (parámetros –i y – I)

- Con **–i** se especifican el número de bandas
- Con **–I** (i maýscula) se especifican el tamaño de la banda (en Kbytes)
- Creamos
  - lvcreate –L 10M –n tercer_lv –i 2 –I 4 mi_vg
    - 10Mbytes en 2 bandas con strip-size de 4 KBytes

```
root@ubsrv64:~#
root@ubsrv64:~# lvcreate  -L 10M -n tercer_vl  -i 2  -I 4  mi_vg
  Rounding up size to full physical extent 12,00 MiB
  Rounding size (3 extents) up to stripe boundary size (4 extents)
  Logical volume "tercer_vl" created
root@ubsrv64:~#
```

15

Creación de VL especificando un PV concreto a utilizar

- lvcreate –L 10M -n cuarto_lv mi_vg /dev/loop1
- Un VL se puede eliminar con **lvremove**

```
root@ubsrv64:~#
root@ubsrv64:~# lvcreate -L 10M -n cuarto_lv mi_vg /dev/loop1
  Rounding up size to full physical extent 12,00 MiB
  Logical volume "cuarto_lv" created
root@ubsrv64:~#
root@ubsrv64:~# _
```

16

Visualización de un VG que ya tiene LV creados

```
root@ubsrv64:~# vgdisplay mi_vg | more
  --- Volume group ---
  VG Name               mi_vg
  System ID
  Format                lvm2
  Metadata Areas        2
  Metadata Sequence No  9
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                4
  Open LV               0
  Max PV                0
  Cur PV                2
  Act PV                2
  VG Size               192,00 MiB
  PE Size               4,00 MiB
  Total PE              48
  Alloc PE / Size       13 / 52,00 MiB
  Free  PE / Size       35 / 140,00 MiB
  VG UUID               UNCe2J-2T3E-ffYb-FGDN-RtM2-9dju-8O6hmW

root@ubsrv64:~#
```

17

Creación de sistemas de ficheros sobre LV de LVM

- Los LV en LVM son equivalentes a las particiones sobre discos físicos
  - Sobre ellos se pueden crear sistemas de ficheros y montarlos sobre directorios
- Creamos 4 directorios de montaje para posteriormente montar los 4 sistemas de ficheros

```
root@ubsrv64:~#
root@ubsrv64:~# mkdir /mnt/primer_lv  /mnt/segundo_lv  /mnt/tercer_lv  /mnt/cuarto_lv
root@ubsrv64:~#
root@ubsrv64:~# _
```

18

Formateamos el primer LV como ext3

- mkfs.ext3 /dev/mi_vg/primer_lv

```
root@ubsrv64:~# mkfs.ext3  /dev/mi_vg/primer_lv
mke2fs 1.42.9 (4-Feb-2014)
Discarding device blocks: hecho
Etiqueta del sistema de ficheros=
OS type: Linux
Tamaño del bloque=1024 (bitácora=0)
Tamaño del fragmento=1024 (bitácora=0)
Stride=0 blocks, Stripe width=0 blocks
3072 inodes, 12288 blocks
614 blocks (5.00%) reserved for the super user
Primer bloque de datos=1
Número máximo de bloques del sistema de ficheros=12582912
2 bloque de grupos
8192 bloques por grupo, 8192 fragmentos por grupo
1536 nodos-i por grupo
Respaldo del superbloque guardado en los bloques:
        8193

Allocating group tables: hecho
Escribiendo las tablas de nodos-i: hecho
Creating journal (1024 blocks): hecho
Escribiendo superbloques y la información contable del sistema de ficheros: hecho

root@ubsrv64:~# _
```

19

El segundo LV lo formateamos como resiserfs

- mkfs.reiserfs --journal-size 513 /dev/mi_vg/segundo_lv

```
root@ubsrv64:~#
root@ubsrv64:~# mkfs.reiserfs  --journal-size 513  /dev/mi_vg/segundo_lv
mkfs.reiserfs 3.6.24

Guessing about desired format.. Kernel 3.13.0-24-generic is running.
Format 3.6 with non-standard journal
Count of blocks on the device: 3072
Number of blocks consumed by mkreiserfs formatting process: 532
Blocksize: 4096
Hash function used to sort names: "r5"
Journal Size 513 blocks (first block 18)
Journal Max transaction length 256
inode generation number: 0
UUID: 8bfb92f7-8ebe-4b43-a960-3152738e3e13
ATTENTION: YOU SHOULD REBOOT AFTER FDISK!
        ALL DATA WILL BE LOST ON '/dev/mi_vg/segundo_lv'!
Continue (y/n):y
Initializing journal - 0%....20%....40%....60%....80%....100%
Syncing..ok
ReiserFS is successfully created on /dev/mi_vg/segundo_lv.
root@ubsrv64:~#
```

20

El tercer volumen lo hacemos jfs

- mkfs.jfs /dev/mi_vg/tercer_lv

```
root@ubsrv64:~# mkfs.jfs  /dev/mi_vg/tercer_vl
El programa «mkfs.jfs» no está instalado. Puede instalarlo escribiendo:
apt-get install jfsutils
root@ubsrv64:~# apt-get install jfsutils
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes NUEVOS:
  jfsutils
0 actualizados, 1 se instalarán, 0 para eliminar y 3 no actualizados.
Necesito descargar 272 kB de archivos.
Se utilizarán 1.094 kB de espacio de disco adicional después de esta operación.
Des:1 http://es.archive.ubuntu.com/ubuntu/ trusty/main jfsutils amd64 1.1.15-2.1 [272 kB]
Descargados 272 kB en 0seg. (288 kB/s)
Seleccionando el paquete jfsutils previamente no seleccionado.
(Leyendo la base de datos ... 58286 ficheros o directorios instalados actualmente.)
Preparing to unpack .../jfsutils_1.1.15-2.1_amd64.deb ...
Unpacking jfsutils (1.1.15-2.1) ...
Processing triggers for man-db (2.6.7.1-1ubuntu1) ...
Configurando jfsutils (1.1.15-2.1) ...
root@ubsrv64:~# mkfs.jfs  /dev/mi_vg/tercer_vl
mkfs.jfs version 1.1.15, 04-Mar-2011
Warning!  All data on device /dev/mi_vg/tercer_vl will be lost!

Continue? (Y/N) Y


Format completed successfully.

16384 kilobytes total disk space.
root@ubsrv64:~#
```

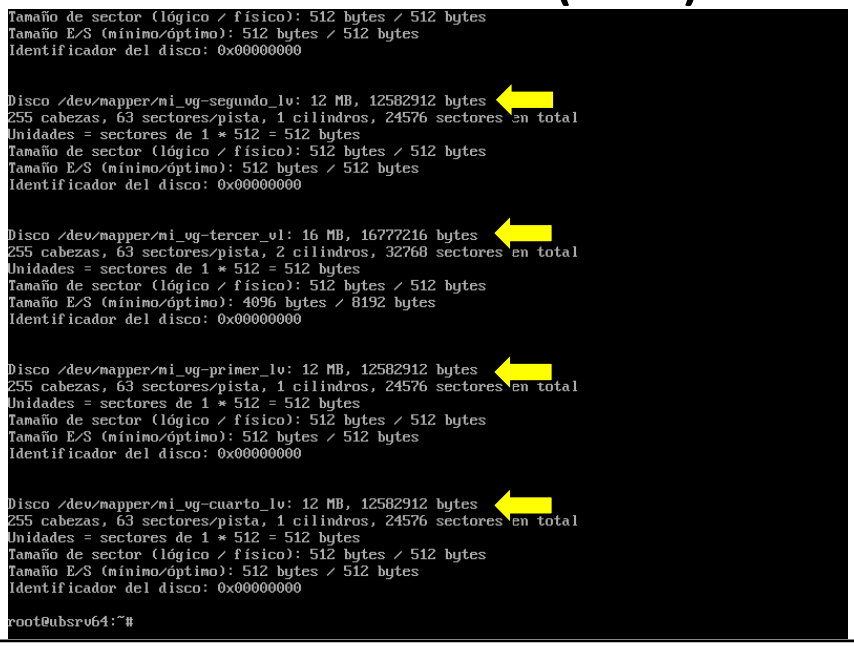
21

Vista de un fdisk –l

```
Disco /dev/sda: 10.7 GB, 10737418240 bytes
255 cabezas, 63 sectores/pista, 1305 cilindros, 20971520 sectores en total
Unidades = sectores de 1 * 512 = 512 bytes
Tamaño de sector (lógico / físico): 512 bytes / 512 bytes
Tamaño E/S (mínimo/óptimo): 512 bytes / 512 bytes
Identificador del disco: 0x000795ce

Dispositivo Inicio    Comienzo      Fin      Bloques  Id  Sistema
/dev/sda1   *        2048      499711      248832   83  Linux
/dev/sda2          501758    20969471    10233857    5  Extendida
/dev/sda5          501760    20969471    10233856   8e  Linux LVM

Disco /dev/mapper/ubsrv64--vg-root: 9403 MB, 9403629568 bytes
255 cabezas, 63 sectores/pista, 1143 cilindros, 18366464 sectores en total
Unidades = sectores de 1 * 512 = 512 bytes
Tamaño de sector (lógico / físico): 512 bytes / 512 bytes
Tamaño E/S (mínimo/óptimo): 512 bytes / 512 bytes
Identificador del disco: 0x00000000


Disco /dev/mapper/ubsrv64--vg-swap_1: 1073 MB, 1073741824 bytes
255 cabezas, 63 sectores/pista, 130 cilindros, 2097152 sectores en total
Unidades = sectores de 1 * 512 = 512 bytes
Tamaño de sector (lógico / físico): 512 bytes / 512 bytes
Tamaño E/S (mínimo/óptimo): 512 bytes / 512 bytes
Identificador del disco: 0x00000000


Disco /dev/mapper/mi_vg-segundo_lv: 12 MB, 12582912 bytes
255 cabezas, 63 sectores/pista, 1 cilindros, 24576 sectores en total
Unidades = sectores de 1 * 512 = 512 bytes
Tamaño de sector (lógico / físico): 512 bytes / 512 bytes
Tamaño E/S (mínimo/óptimo): 512 bytes / 512 bytes
Identificador del disco: 0x00000000

--Más--
```

22

Vista de un fdisk –l (cont.)



23

Extender un LV e instantáneas

- Se puede extender un LV con **lvextend**
- También se pueden hacer instantáneas con la orden **lvcreate -s**
  - Hay que asegurarse de tener cargado el módulo de kernel dm-snapshot
    - Si no lo estuviera: **modprobe dm-snapshot**
  - Se crea la instantánea con:
    - **lvcreate –L20M –s –n congelado /dev/mi_vg/segundo_lv**
  - Que se puede montar con
    - **mkdir /mnt/congelado**
    - **mount /dev/mi_vg/congelado /mnt/congelado/**

24

Operación

- Probar a crear sistemas de ficheros de todo tipo con LVM
- Probar a hacer una instantánea
  - Modificar el sistema de ficheros
  - Comprobar las diferencias entre el sistema de ficheros y la instantánea anterior
- Investigar cómo hacer un espejo y realizarlo
- Realiza un suficiente número de pruebas, tantas como calificación quieras obtener

25

Para entregar

- Ejecución de la práctica
- Pruebas realizadas (describir tanto enunciado de la prueba elegida y su solución)
- ID de práctica:
  - **ISOP509_LVM**

26