→ Convenciones:

```
# En todos los nodos como 'sudo su'.
[root@srv1 ~]# Solo en servidor 'srv1'→ como 'sudo su'.
[root@srv2 ~]# Solo en servidor 'srv2'→ como 'sudo su'.
```

364.3 Advanced LVM (weight: 3)

Weight	3
Description	Candidates should be able to configure LVM volumes. This includes managing LVM snapshot, pools and RAIDs.

Key Knowledge Areas:

- Understand and manage LVM, including linear and striped volumes
- Extend, grow, shrink and move LVM volumes
- Understand and manage LVM snapshots
- Understand and manage LVM thin and thick pools
- Understand and manage LVM RAIDs

Partial list of the used files, terms and utilities:

- /etc/lvm/lvm.conf
- pvcreate
- pvdisplay
- pvmove
- pvremove
- pvresize
- vgcreate
- vgdisplay
- vgreduce
- lvconvert
- lvcreate
- lvdisplay
- lvextend
- lvreduce
- lvresize

→ Logical volumes provide the following advantages over using physical storage directly:

Flexible capacity

When using logical volumes, file systems can extend across multiple disks, since you can aggregate disks and partitions into a single logical volume.

Resizeable storage pools

You can extend logical volumes or reduce logical volumes in size with simple software commands, without reformatting and repartitioning the underlying disk devices.

Online data relocation

To deploy newer, faster, or more resilient storage subsystems, you can move data while your system is active. Data can be rearranged on disks while the disks are in use. For example, you can empty a hot-swappable disk before removing it.

Convenient device naming

Logical storage volumes can be managed in user-defined and custom named groups.

Disk striping

You can create a logical volume that stripes data across two or more disks. This can dramatically increase throughput.

Mirroring volumes

Logical volumes provide a convenient way to configure a mirror for your data.

Volume snapshots

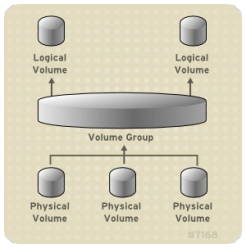
Using logical volumes, you can take device snapshots for consistent backups or to test the effect of changes without affecting the real data.

Thin volumes

Logical volumes can be thinly provisioned. This allows you to create logical volumes that are larger than the available extents.

Cache volumes

A cache logical volume uses a small logical volume consisting of fast block devices (such as SSD drives) to improve the performance of a larger and slower logical volume by storing the frequently used blocks on the smaller, faster logical volume.



1. Physical volumes

The underlying physical storage unit of an LVM logical volume is a block device such as a partition

or whole disk. To use the device for an LVM logical volume, the device must be initialized as a physical volume (PV). Initializing a block device as a physical volume places a label near the start of the device.

By default, the LVM label is placed in the second 512-byte sector. You can overwrite this default by placing the label on any of the first 4 sectors when you create the physical volume. This allows LVM volumes to co-exist with other users of these sectors, if necessary.

An LVM label provides correct identification and device ordering for a physical device, since devices can come up in any order when the system is booted. An LVM label remains persistent across reboots and throughout a cluster.

The LVM label identifies the device as an LVM physical volume. It contains a random unique identifier (the UUID) for the physical volume. It also stores the size of the block device in bytes, and it records where the LVM metadata will be stored on the device.

The LVM metadata contains the configuration details of the LVM volume groups on your system. By default, an identical copy of the metadata is maintained in every metadata area in every physical volume within the volume group. LVM metadata is small and stored as ASCII.

Currently LVM allows you to store 0, 1 or 2 identical copies of its metadata on each physical volume. The default is 1 copy. Once you configure the number of metadata copies on the physical volume, you cannot change that number at a later time. The first copy is stored at the start of the device, shortly after the label. If there is a second copy, it is placed at the end of the device. If you accidentally overwrite the area at the beginning of your disk by writing to a different disk than you intend, a second copy of the metadata at the end of the device will allow you to recover the metadata.

2. Volume groups

Physical volumes are combined into volume groups (VGs). This creates a pool of disk space out of which logical volumes can be allocated.

Within a volume group, the disk space available for allocation is divided into units of a fixed-size called extents. An extent is the smallest unit of space that can be allocated. Within a physical volume, extents are referred to as physical extents.

A logical volume is allocated into logical extents of the same size as the physical extents. The extent size is thus the same for all logical volumes in the volume group. The volume group maps the logical extents to physical extents.

3. LVM logical volumes

In LVM, a volume group is divided up into logical volumes. An administrator can grow or shrink logical volumes without destroying data, unlike standard disk partitions. If the physical volumes in

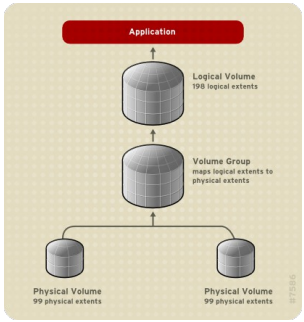
a volume group are on separate drives or RAID arrays, then administrators can also spread a logical volume across the storage devices.

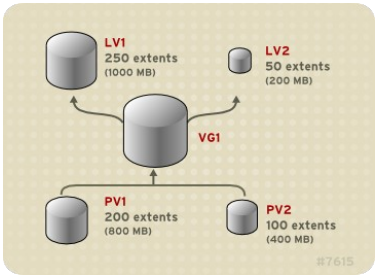
You can lose data if you shrink a logical volume to a smaller capacity than the data on the volume requires. To ensure maximum flexibility, create logical volumes to meet your current needs, and leave excess storage capacity unallocated. You can safely extend logical volumes to use unallocated space, depending on your needs.

3.1. Linear Volumes

A linear volume aggregates space from one or more physical volumes into one logical volume. For example, if you have two 60GB disks, you can create a 120GB logical volume. The physical storage is concatenated.

Creating a linear volume assigns a range of physical extents to an area of a logical volume in order. For example, logical extents 1 to 99 could map to one physical volume and logical extents 100 to 198 could map to a second physical volume. From the point of view of the application, there is one device that is 198 extents in size.





3.2. Striped Logical Volumes

When you write data to an LVM logical volume, the file system lays the data out across the underlying physical volumes. You can control the way the data is written to the physical volumes by

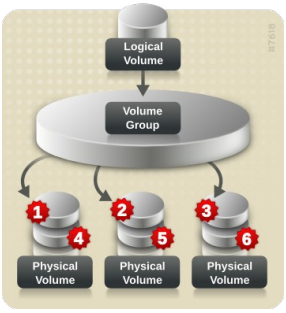
creating a striped logical volume. For large sequential reads and writes, this can improve the efficiency of the data I/O.

Striping enhances performance by writing data to a predetermined number of physical volumes in round-robin fashion. With striping, I/O can be done in parallel. In some situations, this can result in near-linear performance gain for each additional physical volume in the stripe.

The following illustration shows data being striped across three physical volumes. In this figure:

→ the first stripe of data is written to the first physical volume
→ the second stripe of data is written to the second physical volume
→ the third stripe of data is written to the third physical volume
→ the fourth stripe of data is written to the first physical volume

In a striped logical volume, the size of the stripe cannot exceed the size of an extent.



Striped logical volumes can be extended by concatenating another set of devices onto the end of the first set. In order to extend a striped logical volume, however, there must be enough free space on the set of underlying physical volumes that make up the volume group to support the stripe. For example, if you have a two-way stripe that uses up an entire volume group, adding a single physical volume to the volume group will not enable you to extend the stripe. Instead, you must add at least two physical volumes to the volume group.

3.3. RAID logical volumes

LVM supports RAID levels 0, 1, 4, 5, 6, and 10.

An LVM RAID volume has the following characteristics:

→ RAID logical volumes created and managed by LVM leverage the Multiple Devices (MD) kernel drivers.
→ You can temporarily split RAID1 images from the array and merge them back into the array later.
→ LVM RAID volumes support snapshots.

Clusters

RAID logical volumes are not cluster-aware.

Although you can create and activate RAID logical volumes exclusively on one machine, you cannot activate them simultaneously on more than one machine.

Subvolumes

When you create a RAID logical volume, LVM creates a metadata subvolume that is one extent in size for every data or parity subvolume in the array.

For example, creating a 2-way RAID1 array results in two metadata subvolumes (`lv_rmeta_0` and `lv_rmeta_1`) and two data subvolumes (`lv_rimage_0` and `lv_rimage_1`). Similarly, creating a 3-way stripe (plus 1 implicit parity device) RAID4 results in 4 metadata subvolumes (`lv_rmeta_0`, `lv_rmeta_1`, `lv_rmeta_2`, and `lv_rmeta_3`) and 4 data subvolumes (`lv_rimage_0`, `lv_rimage_1`, `lv_rimage_2`, and `lv_rimage_3`).

Integrity

You can lose data when a RAID device fails or when soft corruption occurs. Soft corruption in data storage implies that the data retrieved from a storage device is different from the data written to that device. Adding integrity to a RAID LV helps mitigate or prevent soft corruption.

3.4. Thinly-provisioned logical volumes (thin volumes)

Logical volumes can be thinly provisioned. This allows you to create logical volumes that are larger than the available extents. Using thin provisioning, you can manage a storage pool of free space, known as a thin pool, which can be allocated to an arbitrary number of devices when needed by applications. You can then create devices that can be bound to the thin pool for later allocation when an application actually writes to the logical volume. The thin pool can be expanded dynamically when needed for cost-effective allocation of storage space.

Note

Thin volumes are not supported across the nodes in a cluster. The thin pool and all its thin volumes must be exclusively activated on only one cluster node.

By using thin provisioning, a storage administrator can overcommit the physical storage, often avoiding the need to purchase additional storage. For example, if ten users each request a 100GB file system for their application, the storage administrator can create what appears to be a 100GB file system for each user but which is backed by less actual storage that is used only when needed. When using thin provisioning, it is important that the storage administrator monitor the storage pool and add more capacity if it starts to become full.

To make sure that all available space can be used, LVM supports data discard. This allows for re-use of the space that was formerly used by a discarded file or other block range.

Thin volumes provide support for a new implementation of copy-on-write (COW) snapshot logical volumes, which allow many virtual devices to share the same data in the thin pool.

3.5. Snapshot Volumes

The LVM snapshot feature provides the ability to create virtual images of a device at a particular instant without causing a service interruption. When a change is made to the original device (the origin) after a snapshot is taken, the snapshot feature makes a copy of the changed data area as it was prior to the change so that it can reconstruct the state of the device.

LVM supports thinly-provisioned snapshots.

Because a snapshot copies only the data areas that change after the snapshot is created, the snapshot feature requires a minimal amount of storage. For example, with a rarely updated origin, 3-5 % of the origin's capacity is sufficient to maintain the snapshot.

Snapshot copies of a file system are virtual copies, not an actual media backup for a file system. Snapshots do not provide a substitute for a backup procedure.

The size of the snapshot governs the amount of space set aside for storing the changes to the origin volume. For example, if you made a snapshot and then completely overwrote the origin the snapshot would have to be at least as big as the origin volume to hold the changes. You need to dimension a snapshot according to the expected level of change. So for example a short-lived snapshot of a read-mostly volume, such as `/usr`, would need less space than a long-lived snapshot of a volume that sees a greater number of writes, such as `/home`.

If a snapshot runs full, the snapshot becomes invalid, since it can no longer track changes on the origin volume. You should regularly monitor the size of the snapshot. Snapshots are fully resizable, however, so if you have the storage capacity you can increase the size of the snapshot volume to prevent it from getting dropped. Conversely, if you find that the snapshot volume is larger than you need, you can reduce the size of the volume to free up space that is needed by other logical volumes.

When you create a snapshot file system, full read and write access to the origin stays possible. If a chunk on a snapshot is changed, that chunk is marked and never gets copied from the original volume.

There are several uses for the snapshot feature:

→ Most typically, a snapshot is taken when you need to perform a backup on a logical volume without halting the live system that is continuously updating the data.
→ You can execute the `fsck` command on a snapshot file system to check the file system integrity and determine whether the original file system requires file system repair.
→ Because the snapshot is read/write, you can test applications against production data by taking a snapshot and running tests against the snapshot, leaving the real data untouched.
→ You can create LVM volumes for use with Red Hat Virtualization. LVM snapshots can be used to create snapshots of virtual guest images. These snapshots can provide a convenient

way to modify existing guests or create new guests with minimal additional storage.

You can use the `--merge` option of the `lvconvert` command to merge a snapshot into its origin volume. One use for this feature is to perform system rollback if you have lost data or files or otherwise need to restore your system to a previous state. After you merge the snapshot volume, the resulting logical volume will have the origin volume's name, minor number, and UUID and the merged snapshot is removed.

3.6. Thinly-provisioned snapshot volumes

Red Hat Enterprise Linux provides support for thinly-provisioned snapshot volumes. Thin snapshot volumes allow many virtual devices to be stored on the same data volume. This simplifies administration and allows for the sharing of data between snapshot volumes.

As for all LVM snapshot volumes, as well as all thin volumes, thin snapshot volumes are not supported across the nodes in a cluster. The snapshot volume must be exclusively activated on only one cluster node.

Thin snapshot volumes provide the following benefits:

→ A thin snapshot volume can reduce disk usage when there are multiple snapshots of the same origin volume.
→ If there are multiple snapshots of the same origin, then a write to the origin will cause one COW operation to preserve the data. Increasing the number of snapshots of the origin should yield no major slowdown.
→ Thin snapshot volumes can be used as a logical volume origin for another snapshot. This allows for an arbitrary depth of recursive snapshots (snapshots of snapshots of snapshots…).
→ A snapshot of a thin logical volume also creates a thin logical volume. This consumes no data space until a COW operation is required, or until the snapshot itself is written.
→ A thin snapshot volume does not need to be activated with its origin, so a user may have only the origin active while there are many inactive snapshot volumes of the origin.
→ When you delete the origin of a thinly-provisioned snapshot volume, each snapshot of that origin volume becomes an independent thinly-provisioned volume. This means that instead of merging a snapshot with its origin volume, you may choose to delete the origin volume and then create a new thinly-provisioned snapshot using that independent volume as the origin volume for the new snapshot.

Although there are many advantages to using thin snapshot volumes, there are some use cases for which the older LVM snapshot volume feature may be more appropriate to your needs:

→ You cannot change the chunk size of a thin pool. If the thin pool has a large chunk size (for example, 1MB) and you require a short-living snapshot for which a chunk size that large is not efficient, you may elect to use the older snapshot feature.
→ You cannot limit the size of a thin snapshot volume; the snapshot will use all of the space

in the thin pool, if necessary. This may not be appropriate for your needs.

In general, you should consider the specific requirements of your site when deciding which snapshot format to use.

3.7. Cache Volumes

LVM supports the use of fast block devices (such as SSD drives) as write-back or write-through caches for larger slower block devices. Users can create cache logical volumes to improve the performance of their existing logical volumes or create new cache logical volumes composed of a small and fast device coupled with a large and slow device.

lvm-00 ~ # cat /etc/lvm/lvmlocal.conf

lvm-00 ~ # cat /etc/lvm/lvm.conf

lvm-00 ~ # lsblk /dev/vd[b-z]

```
NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
vdb  252:16   0  20G  0 disk
vdc  252:32   0  20G  0 disk
vdd  252:48   0  20G  0 disk
vde  252:64   0  20G  0 disk
vdf  252:80   0  20G  0 disk
vdg  252:96   0  20G  0 disk
```

→ **Crear PV**

lvm-00 ~ # pvcreate /dev/vd[bcd]

```
  Physical volume "/dev/vdb" successfully created.
  Physical volume "/dev/vdc" successfully created.
  Physical volume "/dev/vdd" successfully created.
```

lvm-00 ~ # pvs /dev/vd[bcd]

```
  PV         VG Fmt  Attr PSize  PFree
  /dev/vdb      lvm2 ---  20,00g 20,00g
  /dev/vdc      lvm2 ---  20,00g 20,00g
  /dev/vdd      lvm2 ---  20,00g 20,00g
```

lvm-00 ~ # pvdisplay /dev/vdb

"/dev/vdb" is a new physical volume of "20,00 GiB"

--- NEW Physical volume ---

PV Name /dev/vdb

VG Name

PV Size 20,00 GiB

Allocatable NO

PE Size 0

Total PE 0

Free PE 0

Allocated PE 0

PV UUID bJXb1f-o0Uu-WTLq-QhFN-3vfD-VsjQ-WGIlwv

→ **Crear VG**

lvm-00 ~ # vgcreate vg_01 /dev/vd[bc]

Volume group "vg_01" successfully created

lvm-00 ~ # vgs vg_01

VG #PV #LV #SN Attr VSize VFree

vg_01 2 0 0 wz--n- 39,99g 39,99g

lvm-00 ~ # pvs /dev/vd[bc]

PV VG Fmt Attr PSize PFree

/dev/vdb vg_01 lvm2 a-- <20,00g <20,00g

/dev/vdc vg_01 lvm2 a-- <20,00g <20,00g

lvm-00 ~ # vgdisplay vg_01

--- Volume group ---

VG Name vg_01

System ID

Format lvm2

Metadata Areas 2

Metadata Sequence No 1

```
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                0
  Open LV               0
  Max PV                0
  Cur PV                2
  Act PV                2
  VG Size               39,99 GiB
  PE Size               4,00 MiB
  Total PE              10238
  Alloc PE / Size       0 / 0
  Free  PE / Size       10238 / 39,99 GiB
  VG UUID               MLiI2m-adjD-ethC-oDxk-wM8F-v6bj-6K27FO
lvm-00 ~ # vgextend vg_01 /dev/vdd
  Volume group "vg_01" successfully extended
lvm-00 ~ # vgs vg_01
  VG    #PV #LV #SN Attr   VSize   VFree
  vg_01   3   0   0 wz--n- <59,99g <59,99g
lvm-00 ~ # pvs /dev/vd[bcd]
  PV         VG    Fmt  Attr PSize   PFree
  /dev/vdb   vg_01 lvm2 a--  <20,00g <20,00g
  /dev/vdc   vg_01 lvm2 a--  <20,00g <20,00g
  /dev/vdd   vg_01 lvm2 a--  <20,00g <20,00g
```

→ **Crear LV**

```
lvm-00 ~ # lvcreate -n lv_01 -L 500M vg_01
lvm-00 ~ # lvdisplay /dev/vg_01/lv_01
  --- Logical volume ---
```

```
  LV Path                /dev/vg_01/lv_01
  LV Name                lv_01
  VG Name                vg_01
  LV UUID                7y1PAX-VKj8-BHKd-k2mt-Twle-jGZ0-g1qUG6
  LV Write Access        read/write
  LV Creation host, time lvm-00.cadilinea.com, 2021-03-28 20:36:16 +0200
  LV Status              available
  # open                 0
  LV Size                500,00 MiB
  Current LE             125
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     8192
  Block device           253:6
```

→ Crear FS

```
lvm-00 ~ # mkfs.xfs /dev/vg_01/lv_01
meta-data=/dev/vg_01/lv_01       isize=512    agcount=4, agsize=32000 blks
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=1        finobt=1, sparse=1, rmapbt=0
         =                       reflink=1
data     =                       bsize=4096   blocks=128000, imaxpct=25
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0, ftype=1
log      =internal log           bsize=4096   blocks=1368, version=2
         =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
Discarding blocks...Done.
```

```
lvm-00 ~ # mount /dev/vg_01/lv_01 /mnt/
lvm-00 ~ # df /dev/vg_01/lv_01
S.ficheros              bloques de 1K Usados Disponibles Uso% Montado en
/dev/mapper/vg_01-lv_01       506528  29388      477140   6% /mnt
lvm-00 ~ # lsblk /dev/vg_01/lv_01 -f
NAME       FSTYPE LABEL UUID                                 MOUNTPOINT
vg_01-lv_01 xfs         d1ff35c0-2494-4815-ae88-05bf54caa834 /mnt
lvm-00 ~ # lsblk /dev/mapper/vg_01-lv_01 -f
NAME       FSTYPE LABEL UUID                                 MOUNTPOINT
vg_01-lv_01 xfs         d1ff35c0-2494-4815-ae88-05bf54caa834 /mnt
```

→ CREATING A RAID0 (STRIPED) LOGICAL VOLUME

```
lvm-00 ~ # lvremove /dev/vg_01/lv_01
Do you really want to remove active logical volume vg_01/lv_01? [y/n]: y
  Logical volume "lv_01" successfully removed
lvm-00 ~ # vgremove vg_01
  Volume group "vg_01" successfully removed
lvm-00 ~ # pvremove /dev/vd[bcd]
  Labels on physical volume "/dev/vdb" successfully wiped.
  Labels on physical volume "/dev/vdc" successfully wiped.
  Labels on physical volume "/dev/vdd" successfully wiped.
lvm-00 ~ # pvcreate /dev/vd[bcd]
  Physical volume "/dev/vdb" successfully created.
  Physical volume "/dev/vdc" successfully created.
  Physical volume "/dev/vdd" successfully created.
lvm-00 ~ # vgcreate vg_raid0 /dev/vd[bcd]
  Volume group "vg_raid0" successfully created
lvm-00 ~ # vgs vg_raid0
```

```
  VG       #PV #LV #SN Attr   VSize   VFree
  vg_raid0   3   0   0 wz--n- <59,99g <59,99g
lvm-00 ~ # lvcreate -n lv_raid0 --type raid0 --stripes 3 --stripesize 4 -L 2G vg_raid0
  Rounding size 2,00 GiB (512 extents) up to stripe boundary size 2,00 GiB(513 extents).
WARNING: xfs signature detected on /dev/vg_raid0/lv_raid0 at offset 0. Wipe it? [y/n]: y
  Wiping xfs signature on /dev/vg_raid0/lv_raid0.
  Logical volume "lv_raid0" created.
lvm-00 ~ # mkfs.ext4 /dev/vg_raid0/lv_raid0
mke2fs 1.45.6 (20-Mar-2020)
Descartando los bloques del dispositivo: hecho
Se está creando un sistema de ficheros con 525312 bloques de 4k y 131376 nodos-i
UUID del sistema de ficheros: 47773046-039d-45e2-b2f9-8bc8c7e12f0b
Respaldos del superbloque guardados en los bloques:
        32768, 98304, 163840, 229376, 294912

Reservando las tablas de grupo: hecho
Escribiendo las tablas de nodos-i: hecho
Creando el fichero de transacciones (16384 bloques): hecho
Escribiendo superbloques y la información contable del sistema de ficheros: hecho
lvm-00 ~ # mount /dev/vg_raid0/lv_raid0 /mnt/
lvm-00 ~ # df /dev/vg_raid0/lv_raid0 -hT
S.ficheros                Tipo Tamaño Usados  Disp Uso% Montado en
/dev/mapper/vg_raid0-lv_raid0 ext4   2,0G   6,1M  1,8G   1% /mnt
lvm-00 ~ # lsblk -f
…
vdb                       LVM2_member       OlVq63-vknn-gLPq-krvU-1FBd-5T4j-TndBT4
└─vg_raid0-lv_raid0_rimage_0
  └─vg_raid0-lv_raid0     ext4              47773046-039d-45e2-b2f9-8bc8c7e12f0b   /mnt
```

```
vdc                    LVM2_member      jTBHz7-7qE3-nSzo-7AbE-Pya9-x7PT-KwNtOs
└─vg_raid0-lv_raid0_rimage_1
  └─vg_raid0-lv_raid0  ext4             47773046-039d-45e2-b2f9-8bc8c7e12f0b   /mnt
vdd                    LVM2_member      dnDVhV-i8M7-TwrX-IhoJ-VgVp-hH6d-p2zEIH
└─vg_raid0-lv_raid0_rimage_2
  └─vg_raid0-lv_raid0  ext4             47773046-039d-45e2-b2f9-8bc8c7e12f0b   /mnt
```

→ RENAMING LVM LOGICAL VOLUMES

(If the logical volume exists in a clustered environment, deactivate the logical volume on all nodes where it is active. Use the following command on each such node:

 # lvchange --activate n vg-name/lv-name).

```
lvm-00 ~ # umount /mnt
lvm-00 ~ # lvrename /dev/vg_raid0/lv_raid0 /dev/vg_raid0/lv_raid0-renombrado
  Renamed "lv_raid0" to "lv_raid0-renombrado" in volume group "vg_raid0"
lvm-00 ~ # mount /dev/vg_raid0/lv_raid0-renombrado /mnt/
lvm-00 ~ # df /dev/vg_raid0/lv_raid0-renombrado
S.ficheros                          bloques de 1K Usados Disponibles Uso% Montado en
/dev/mapper/vg_raid0-lv_raid0--renombrado      2002684   6168     1875072   1% /mnt
```

→REMOVING A DISK FROM A LOGICAL VOLUME

```
lvm-00 ~ # pvs -o+pv_used /dev/vd[b-d]
  PV         VG       Fmt  Attr PSize   PFree   Used
  /dev/vdb   vg_raid0 lvm2 a--  <20,00g <19,33g 684,00m
  /dev/vdc   vg_raid0 lvm2 a--  <20,00g <19,33g 684,00m
  /dev/vdd   vg_raid0 lvm2 a--  <20,00g <19,33g 684,00m
lvm-00 ~ # pvmove /dev/vdd
  Insufficient free space: 171 extents needed, but only 0 available
  Unable to allocate mirror extents for vg_raid0/pvmove0.
  Failed to convert pvmove LV to mirrored.
```

```
lvm-00 ~ # pvdisplay /dev/vd[b-d] |grep 'Free PE'
  Free PE               4948
  Free PE               4948
  Free PE               4948
lvm-00 ~ # umount /mnt
lvm-00 ~ # lvremove /dev/vg_raid0/lv_raid0-renombrado
Do you really want to remove active logical volume vg_raid0/lv_raid0-renombrado? [y/n]: y
  Logical volume "lv_raid0-renombrado" successfully removed
lvm-00 ~ # lvcreate -n lv_raid0 --type raid0 --stripes 3 --stripesize 4 -l 100%FREE vg_raid0
WARNING: ext4 signature detected on /dev/vg_raid0/lv_raid0 at offset 1080. Wipe it? [y/n]: y
  Wiping ext4 signature on /dev/vg_raid0/lv_raid0.
  Logical volume "lv_raid0" created.
lvm-00 ~ # pvs -o+pv_used /dev/vd[b-d]
  PV         VG       Fmt  Attr PSize   PFree Used
  /dev/vdb   vg_raid0 lvm2 a--  <20,00g    0  <20,00g
  /dev/vdc   vg_raid0 lvm2 a--  <20,00g    0  <20,00g
  /dev/vdd   vg_raid0 lvm2 a--  <20,00g    0  <20,00g
lvm-00 ~ # vgextend vg_raid0 /dev/vde
  Physical volume "/dev/vde" successfully created.
  Volume group "vg_raid0" successfully extended

lvm-00 ~ # pvs -o+pv_used /dev/vd[b-e]
  PV         VG       Fmt  Attr PSize   PFree   Used
  /dev/vdb   vg_raid0 lvm2 a--  <20,00g     0  <20,00g
  /dev/vdc   vg_raid0 lvm2 a--  <20,00g     0  <20,00g
  /dev/vdd   vg_raid0 lvm2 a--  <20,00g     0  <20,00g
  /dev/vde   vg_raid0 lvm2 a--  <20,00g <20,00g      0
```

```
lvm-00 ~ # pvmove /dev/vdd
  /dev/vdd: Moved: 1,17%
…
  /dev/vdd: Moved: 100,00%
lvm-00 ~ # pvs -o+pv_used /dev/vd[b-e]
  PV         VG       Fmt  Attr PSize   PFree   Used
  /dev/vdb   vg_raid0 lvm2 a--  <20,00g      0  <20,00g
  /dev/vdc   vg_raid0 lvm2 a--  <20,00g      0  <20,00g
  /dev/vdd   vg_raid0 lvm2 a--  <20,00g <20,00g      0
  /dev/vde   vg_raid0 lvm2 a--  <20,00g      0  <20,00g
lvm-00 ~ # vgreduce vg_raid0 /dev/vdd
  Removed "/dev/vdd" from volume group "vg_raid0"
lvm-00 ~ # pvs /dev/vd[b-e]
  PV         VG       Fmt  Attr PSize   PFree
  /dev/vdb   vg_raid0 lvm2 a--  <20,00g     0
  /dev/vdc   vg_raid0 lvm2 a--  <20,00g     0
  /dev/vdd            lvm2 ---   20,00g 20,00g
  /dev/vde   vg_raid0 lvm2 a--  <20,00g     0
```

→ Moving Extents to a New Disk

```
lvm-00 ~ # vgextend vg_raid0 /dev/vdd
  Volume group "vg_raid0" successfully extended
lvm-00 ~ # pvmove /dev/vde0 /dev/vdd
  /dev/vde: Moved: 0,04%
…
  /dev/vde: Moved: 100,00%
lvm-00 ~ # pvs /dev/vd[b-e]
  PV         VG       Fmt  Attr PSize   PFree
  /dev/vdb   vg_raid0 lvm2 a--  <20,00g      0
```

```
  /dev/vdc   vg_raid0 lvm2 a--  <20,00g      0
  /dev/vdd   vg_raid0 lvm2 a--  <20,00g      0
  /dev/vde   vg_raid0 lvm2 a--  <20,00g <20,00g
lvm-00 ~ # vgreduce vg_raid0 /dev/vde
  Removed "/dev/vde" from volume group "vg_raid0"
lvm-00 ~ # vgs vg_raid0
  VG       #PV #LV #SN Attr   VSize   VFree
  vg_raid0   3   1   0 wz--n- <59,99g    0
lvm-00 ~ # pvs
  PV         VG       Fmt  Attr PSize   PFree
…
  /dev/vdb   vg_raid0 lvm2 a--  <20,00g     0
  /dev/vdc   vg_raid0 lvm2 a--  <20,00g     0
  /dev/vdd   vg_raid0 lvm2 a--  <20,00g     0
  /dev/vde           lvm2 ---   20,00g 20,00g
```

→REMOVING LVM LOGICAL VOLUMES

1. If the logical volume is currently mounted, unmount the volume.

2. If the logical volume exists in a clustered environment, deactivate the logical volume on all nodes where it is active. Use the following command on each such node:

 # lvchange --activate n vg-name/lv-name

3. Remove the logical volume using the lvremove utility:

 # lvremove /dev/vg-name/lv-name

```
Do you really want to remove active logical volume "lv-name"? [y/n]: y
Logical volume "lv-name" successfully removed
lvm-00 ~ # lvs
  LV       VG       Attr       LSize   Pool   Origin Data%  Meta%  Move Log Cpy%Sync Convert
  …
  lv_raid0 vg_raid0 rwi-a-r--- <59,99g
```

lvm-00 ~ # umount /mnt

lvm-00 ~ # lvremove /dev/vg_raid0/lv_raid0

Do you really want to remove active logical volume vg_raid0/lv_raid0? [y/n]: y

Logical volume "lv_raid0" successfully removed

→ GROWING LOGICAL VOLUMES

Por tamaño:

lvm-00 ~ # vgcreate vg_test /dev/vd[bcd]

lvm-00 ~ # lvcreate -n lv_test -L 1G vg_test

lvm-00 ~ # lvs /dev/vg_test/lv_test

LV VG Attr LSize Pool Origin Data% Meta% Move Log Cpy%Sync Convert

lv_test vg_test -wi-a----- 1,00g

lvm-00 ~ # lvextend -L 2G /dev/vg_test/lv_test

Size of logical volume vg_test/lv_test changed from 1,00 GiB (256 extents) to 2,00 GiB (512 extents).

Logical volume vg_test/lv_test successfully resized.

lvm-00 ~ # lvs /dev/vg_test/lv_test

LV VG Attr LSize Pool Origin Data% Meta% Move Log Cpy%Sync Convert

lv_test vg_test -wi-a----- 2,00g

lvm-00 ~ # lvextend -L +2G /dev/vg_test/lv_test

Size of logical volume vg_test/lv_test changed from 2,00 GiB (512 extents) to 4,00 GiB (1024 extents).

Logical volume vg_test/lv_test successfully resized.

lvm-00 ~ # lvs /dev/vg_test/lv_test

LV VG Attr LSize Pool Origin Data% Meta% Move Log Cpy%Sync Convert

lv_test vg_test -wi-a----- 4,00g

Por extents:

lvm-00 ~ # lvremove /dev/vg_test/lv_test

lvm-00 ~ # lvcreate -n lv_test -l 100 vg_test

lvm-00 ~ # lvs /dev/vg_test/lv_test

```
  LV      VG      Attr       LSize   Pool Origin Data%  Meta%  Move Log Cpy%Sync Convert
  lv_test vg_test -wi-a----- 400,00m
```

lvm-00 ~ # lvextend /dev/vg_test/lv_test -l 200

```
  Size of logical volume vg_test/lv_test changed from 400,00 MiB (100 extents) to 800,00 MiB (200 extents).
  Logical volume vg_test/lv_test successfully resized.
```

lvm-00 ~ # lvextend /dev/vg_test/lv_test -l +200

```
  Size of logical volume vg_test/lv_test changed from 800,00 MiB (200 extents) to 1,56 GiB (400 extents).
  Logical volume vg_test/lv_test successfully resized.
```

lvm-00 ~ # lvextend /dev/vg_test/lv_test -l 100

```
  New size given (100 extents) not larger than existing size (400 extents)
```

lvm-00 ~ # lvreduce -l 100 /dev/vg_test/lv_test

```
  WARNING: Reducing active logical volume to 400,00 MiB.
  THIS MAY DESTROY YOUR DATA (filesystem etc.)
Do you really want to reduce vg_test/lv_test? [y/n]: y
  Size of logical volume vg_test/lv_test changed from 1,56 GiB (400 extents) to 400,00 MiB (100 extents).
  Logical volume vg_test/lv_test successfully resized.
```

→SHRINKING LOGICAL VOLUMES

Shrinking is not supported on a GFS2 or XFS file system, so you cannot reduce the size of a logical volume that contains a GFS2 or XFS file system.

If the logical volume you are reducing contains a file system, to prevent data loss you must ensure that the file system is not using the space in the logical volume that is being reduced. For this reason, it is recommended that you use the --resizefs option of the lvreduce command when the logical volume contains a file system. When you use this option, the lvreduce command attempts to reduce the file system before shrinking the logical volume. If shrinking the file system fails, as can occur if the file system is full or the file system does not support shrinking, then the lvreduce command will fail and not attempt to shrink the logical volume.

lvm-00 ~ # lvs /dev/vg_test/lv_test

```
  LV      VG      Attr       LSize   Pool Origin Data%  Meta%  Move Log Cpy%Sync Convert
```

lv_test vg_test -wi-a----- 400,00m

lvm-00 ~ # mkfs.ext4 /dev/vg_test/lv_test

mke2fs 1.45.6 (20-Mar-2020)

Descartando los bloques del dispositivo: hecho

Se está creando un sistema de ficheros con 409600 bloques de 1k y 102400 nodos-i

UUID del sistema de ficheros: a9b05d56-cfcf-4375-bbb4-8cee2369fa4e

Respaldos del superbloque guardados en los bloques:

8193, 24577, 40961, 57345, 73729, 204801, 221185, 401409

Reservando las tablas de grupo: hecho

Escribiendo las tablas de nodos-i: hecho

Creando el fichero de transacciones (8192 bloques): hecho

Escribiendo superbloques y la información contable del sistema de ficheros: hecho

lvm-00 ~ # lsblk /dev/vdb -f

NAME FSTYPE LABEL UUID MOUNTPOINT

vdb LVM2_member OlVq63-vknn-gLPq-krvU-1FBd-5T4j-TndBT4

└─vg_test-lv_test ext4 a9b05d56-cfcf-4375-bbb4-8cee2369fa4e

lvm-00 ~ # df -hT /dev/vg_test/lv_test

S.ficheros Tipo Tamaño Usados Disp Uso% Montado en

devtmpfs devtmpfs 1,4G 0 1,4G 0% /dev

lvm-00 ~ # mount /dev/vg_test/lv_test /mnt/

lvm-00 ~ # touch /mnt/fichero_test.txt

lvm-00 ~ # lvresize --resizefs -L 200M /dev/vg_test/lv_test

Do you want to unmount "/mnt" ? [Y|n] y

fsck de util-linux 2.32.1

/dev/mapper/vg_test-lv_test: 12/102400 ficheros (0.0% no contiguos), 23457/409600 bloques

resize2fs 1.45.6 (20-Mar-2020)

Cambiando el tamaño del sistema de ficheros en /dev/mapper/vg_test-lv_test a 204800 (1k)

bloques.

El sistema de ficheros en /dev/mapper/vg_test-lv_test tiene ahora 204800 bloques (de 1k).

Size of logical volume vg_test/lv_test changed from 400,00 MiB (100 extents) to 200,00 MiB (50 extents).

Logical volume vg_test/lv_test successfully resized.

lvm-00 ~ # lvs /dev/vg_test/lv_test

LV VG Attr LSize Pool Origin Data% Meta% Move Log Cpy%Sync Convert

lv_test vg_test -wi-ao---- 200,00m

lvm-00 ~ # ls /mnt/

fichero_test.txt lost+found

lvm-00 ~ # resize2fs /dev/vg_test/lv_test

resize2fs 1.45.6 (20-Mar-2020)

El sistema de ficheros ya tiene 204800 bloques (1k) de longitud. ¡No hay que hacer nada!

lvm-00 ~ # lvresize -L +200M /dev/vg_test/lv_test

Size of logical volume vg_test/lv_test changed from 200,00 MiB (50 extents) to 400,00 MiB (100 extents).

Logical volume vg_test/lv_test successfully resized.

lvm-00 ~ # resize2fsM /dev/vg_test/lv_test

resize2fs 1.45.6 (20-Mar-2020)

El sistema de ficheros de /dev/vg_test/lv_test está montado en /mnt; hace falta cambiar el tamaño en línea

old_desc_blocks = 2, new_desc_blocks = 4

El sistema de ficheros en /dev/vg_test/lv_test tiene ahora 409600 bloques (de 1k).

lvm-00 ~ # lvs /dev/vg_test/lv_test

LV VG Attr LSize Pool Origin Data% Meta% Move Log Cpy%Sync Convert

lv_test vg_test -wi-ao---- 400,00m

lvm-00 ~ # ls /mnt/

fichero_test.txt lost+found

→ pvresize

```
lvm-00 ~ # pvs /dev/vd[b-e]
  PV         VG      Fmt  Attr PSize   PFree
  /dev/vdb   vg_test lvm2 a--  <20,00g <19,61g
  /dev/vdc   vg_test lvm2 a--  <20,00g <20,00g
  /dev/vdd   vg_test lvm2 a--  <20,00g <20,00g
  /dev/vde           lvm2 ---   20,00g  20,00g
lvm-00 ~ # pvresize --setphysicalvolumesize 10G /dev/vde
/dev/vde: Requested size 10,00 GiB is less than real size 20,00 GiB. Proceed?  [y/n]: y
  WARNING: /dev/vde: Pretending size is 20971520 not 41943040 sectors.
  Physical volume "/dev/vde" changed
  1 physical volume(s) resized or updated / 0 physical volume(s) not resized
lvm-00 ~ # ps /dev/vd[b-e]
  PV         VG      Fmt  Attr PSize   PFree
  /dev/vdb   vg_test lvm2 a--  <20,00g <19,61g
  /dev/vdc   vg_test lvm2 a--  <20,00g <20,00g
  /dev/vdd   vg_test lvm2 a--  <20,00g <20,00g
  /dev/vde           lvm2 ---   10,00g  10,00g
lvm-00 ~ # pvresize --setphysicalvolumesize 20G /dev/vde
  Physical volume "/dev/vde" changed
  1 physical volume(s) resized or updated / 0 physical volume(s) not resized
lvm-00 ~ # pvs /dev/vd[b-e]
  PV         VG      Fmt  Attr PSize   PFree
  /dev/vdb   vg_test lvm2 a--  <20,00g <19,61g
  /dev/vdc   vg_test lvm2 a--  <20,00g <20,00g
  /dev/vdd   vg_test lvm2 a--  <20,00g <20,00g
  /dev/vde           lvm2 ---   20,00g  20,00g
```

===> pendiente snapshots y thick-pools

BIBLIOGRAFIA:

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/5/html/logical_volume_manager_administration/lv_overview#linear_volumes

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/configuring_and_managing_logical_volumes/index

https://web.mit.edu/rhel-doc/5/RHEL-5-manual/Cluster_Logical_Volume_Manager/LV_create.html