# Red Hat Enterprise Linux 8.0 Beta

## Configuring and managing logical volumes

A guide to the configuration and management of LVM logical volumes

Last Updated: 2018-11-16

# Red Hat Enterprise Linux 8.0 Beta Configuring and managing logical volumes

A guide to the configuration and management of LVM logical volumes

## Abstract

This documentation collection provides instructions on how to manage LVM logical volumes on Red Hat Enterprise Linux 8.

# Table of Contents

# THIS IS A BETA VERSION!

Thank you for your interest in Red Hat Enterprise Linux 8.0 Beta. Be aware that:

- Beta code should not be used with production data or on production systems.
- Beta does not include a guarantee of support.
- Feedback and bug reports are welcome. Discussions with your account representative, partner contact, and Technical Account Manager (TAM) are also welcome.
- Upgrades to or from a Beta are not supported or recommended.

# PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

We appreciate your input on our documentation. Please let us know how we could make it better. To do so:

- For simple comments on specific passages, make sure you are viewing the documentation in the Multi-page HTML format. Highlight the part of text that you want to comment on. Then, click the **Add Feedback** pop-up that appears below the highlighted text, and follow the displayed instructions.
- For submitting more complex feedback, create a Bugzilla ticket:
  1. Go to the Bugzilla website.
  2. As the Component, use **Documentation**.
  3. Fill in the **Description** field with your suggestion for improvement. Include a link to the relevant part(s) of documentation.
  4. Click **Submit Bug**.

# CHAPTER 1. LOGICAL VOLUMES

Volume management creates a layer of abstraction over physical storage, allowing you to create logical storage volumes. This provides much greater flexibility in a number of ways than using physical storage directly. In addition, the hardware storage configuration is hidden from the software so it can be resized and moved without stopping applications or unmounting file systems. This can reduce operational costs.

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.

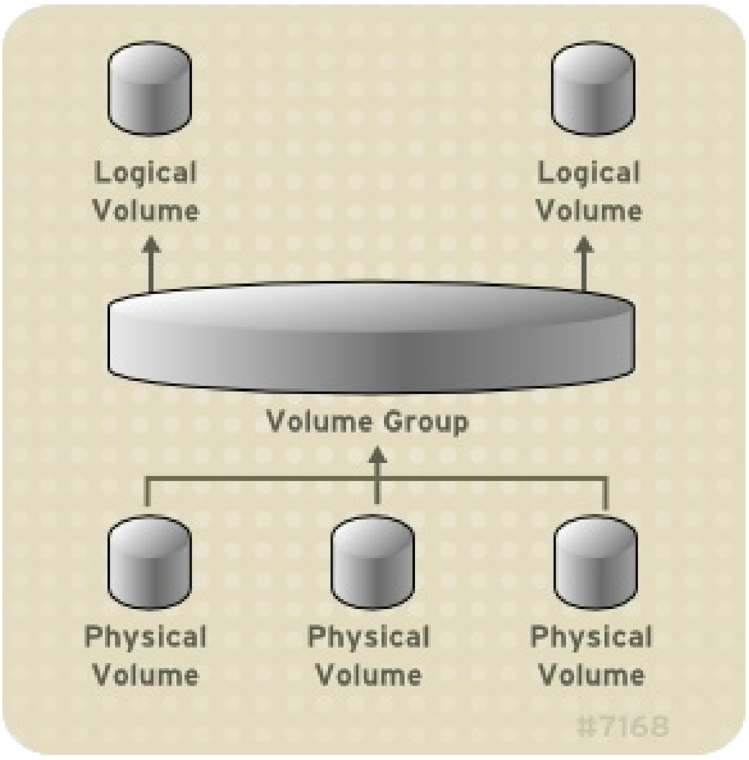
## 1.1. LVM ARCHITECTURE OVERVIEW

The underlying physical storage unit of an LVM logical volume is a block device such as a partition or whole disk. This device is initialized as an LVM *physical volume* (PV).

To create an LVM logical volume, the physical volumes are combined into a *volume group* (VG). This creates a pool of disk space out of which LVM logical volumes (LVs) can be allocated. This process is analogous to the way in which disks are divided into partitions. A logical volume is used by file systems and applications (such as databases).

Figure 1.1, “LVM logical volume components” shows the components of a simple LVM logical volume:

**Figure 1.1. LVM logical volume components**



## 1.2. 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.

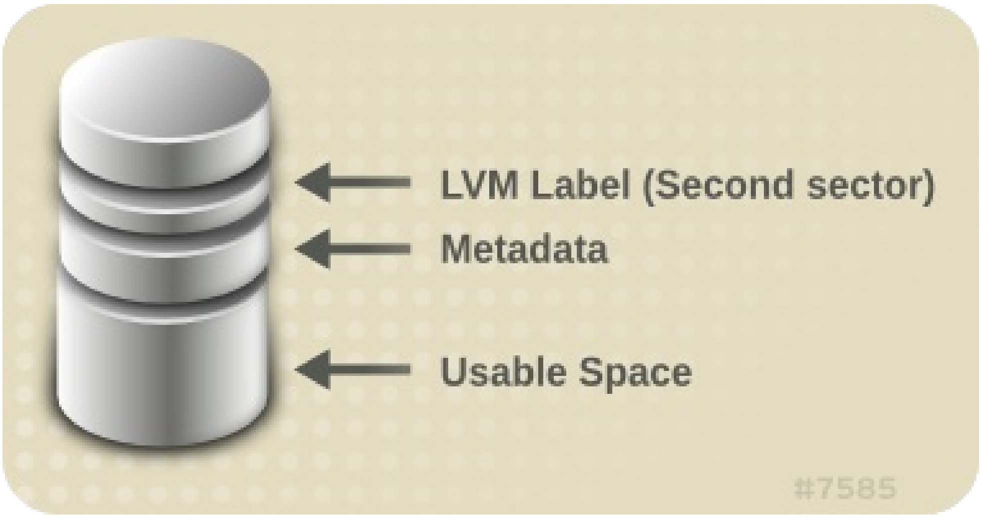
### 1.2.1. LVM physical volume layout

Figure 1.2, “Physical volume layout” shows the layout of an LVM physical volume. The LVM label is on the second sector, followed by the metadata area, followed by the usable space on the device.

**NOTE**

In the Linux kernel (and throughout this document), sectors are considered to be 512 bytes in size.

**Figure 1.2. Physical volume layout**



### 1.2.2. Multiple partitions on a disk

LVM allows you to create physical volumes out of disk partitions. Red Hat recommends that you create a single partition that covers the whole disk to label as an LVM physical volume for the following reasons:

- Administrative convenience
  It is easier to keep track of the hardware in a system if each real disk only appears once. This becomes particularly true if a disk fails. In addition, multiple physical volumes on a single disk may cause a kernel warning about unknown partition types at boot.

- Striping performance
  LVM cannot tell that two physical volumes are on the same physical disk. If you create a striped logical volume when two physical volumes are on the same physical disk, the stripes could be on different partitions on the same disk. This would result in a decrease in performance rather than an increase.

Although it is not recommended, there may be specific circumstances when you will need to divide a disk into separate LVM physical volumes. For example, on a system with few disks it may be necessary to move data around partitions when you are migrating an existing system to LVM volumes. Additionally, if you have a very large disk and want to have more than one volume group for administrative purposes then it is necessary to partition the disk. If you do have a disk with more than one partition and both of those partitions are in the same volume group, take care to specify which partitions are to be included in a logical volume when creating striped volumes.

## 1.3. 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.

## 1.4. LVM LOGICAL VOLUMES

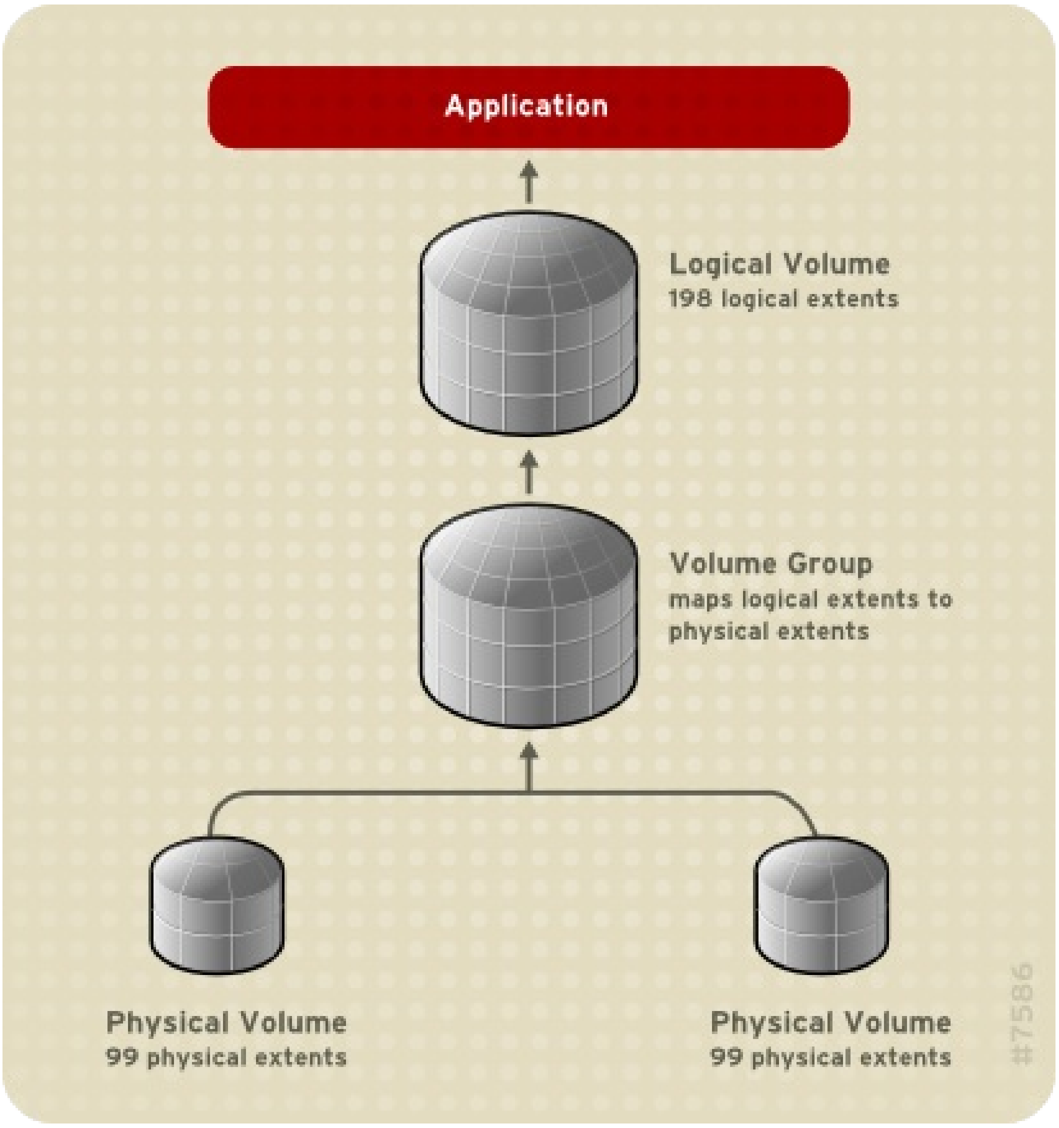
In LVM, a volume group is divided up into logical volumes. The following sections describe the different types of logical volumes.

### 1.4.1. 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.

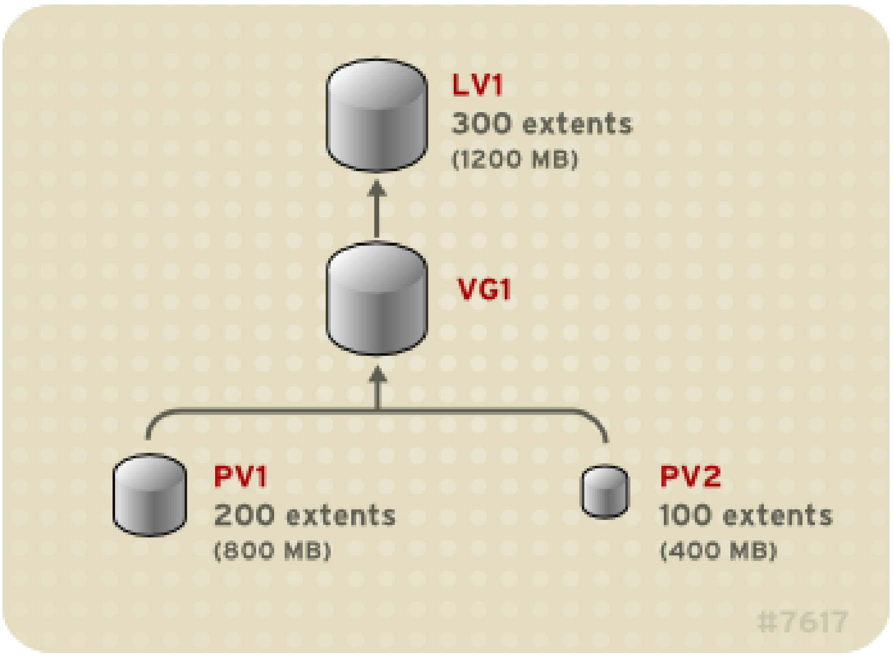
Creating a linear volume assigns a range of physical extents to an area of a logical volume in order. For example, as shown in Figure 1.3, “Extent Mapping” logical extents 1 to 99 could map to one physical volume and logical extents 100 to 198 could map to a second physical volume. From the point of view of the application, there is one device that is 198 extents in size.

**Figure 1.3. Extent Mapping**



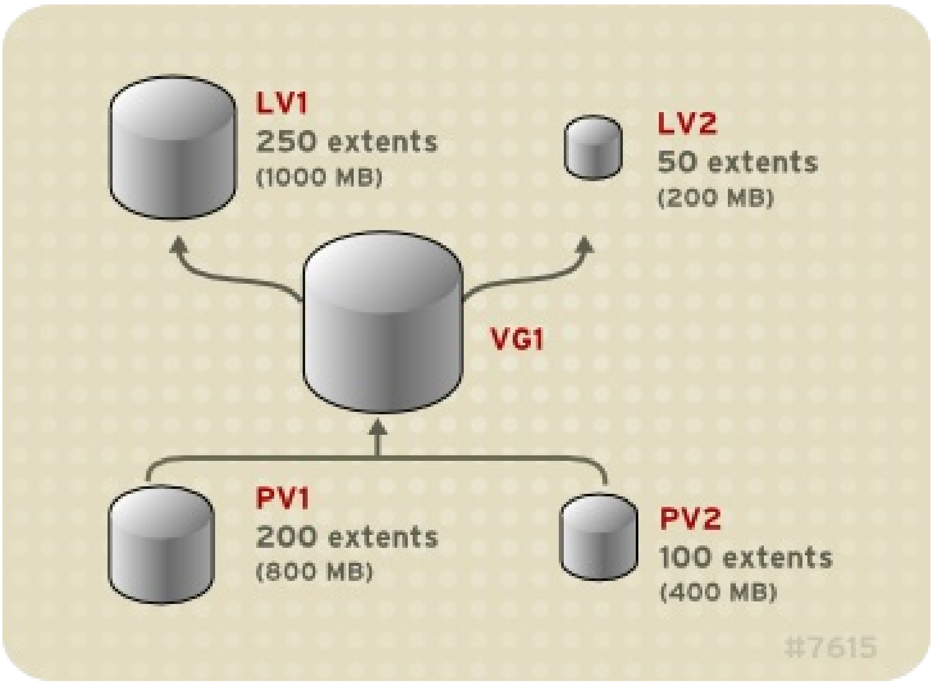
The physical volumes that make up a logical volume do not have to be the same size. Figure 1.4, "Linear volume with unequal physical volumes" shows volume group **VG1** with a physical extent size of 4MB. This volume group includes 2 physical volumes named **PV1** and **PV2**. The physical volumes are divided into 4MB units, since that is the extent size. In this example, **PV1** is 200 extents in size (800MB) and **PV2** is 100 extents in size (400MB). You can create a linear volume any size between 1 and 300 extents (4MB to 1200MB). In this example, the linear volume named **LV1** is 300 extents in size.

**Figure 1.4. Linear volume with unequal physical volumes**



You can configure more than one linear logical volume of whatever size you require from the pool of physical extents. Figure 1.5, “Multiple logical volumes” shows the same volume group as in Figure 1.4, “Linear volume with unequal physical volumes”, but in this case two logical volumes have been carved out of the volume group: **LV1**, which is 250 extents in size (1000MB) and **LV2** which is 50 extents in size (200MB).

**Figure 1.5. Multiple logical volumes**



## 1.4.2. 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.

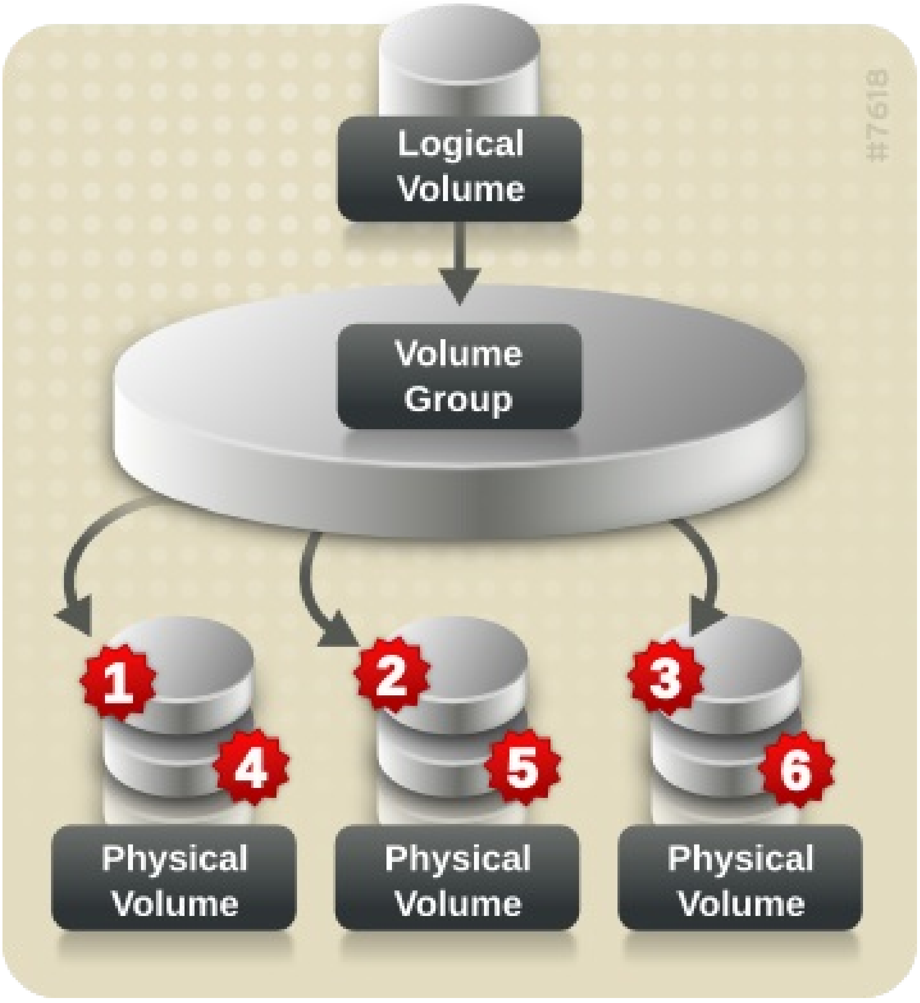
Striping enhances performance by writing data to a predetermined number of physical volumes in round-robin fashion. With striping, I/O can be done in parallel. In some situations, this can result in near-linear performance gain for each additional physical volume in the stripe.

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.

**Figure 1.6. Striping data across three PVs**



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.

### 1.4.3. RAID logical volumes

LVM supports RAID0/1/4/5/6/10. An LVM RAID volume has the following characteristics:

- RAID logical volumes created and managed by means of LVM leverage the MD kernel drivers.
- RAID1 images can be temporarily split from the array and merged back into the array later.
- LVM RAID volumes support snapshots.

**NOTE**

RAID logical volumes are not cluster-aware. While RAID logical volumes can be created and activated exclusively on one machine, they cannot be activated simultaneously on more than one machine. If you require non-exclusive mirrored volumes, you must create the volumes with a `mirror` segment type.

### 1.4.4. 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.

### 1.4.5. 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.

**NOTE**

LVM supports thinly-provisioned snapshots.

**NOTE**

LVM snapshots are not supported across the nodes in a cluster. You cannot create a snapshot volume in a clustered volume group.

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.

**NOTE**

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.

### 1.4.6. 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.

### 1.4.7. 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.

# CHAPTER 2. CONFIGURING LVM LOGICAL VOLUMES

The following procedures provide examples of basic LVM administration tasks.

## 2.1. CREATING AN LVM LOGICAL VOLUME ON THREE DISKS

This example procedure creates an LVM logical volume called **mylv** that consists of the disks at **/dev/sda1**, **/dev/sdb1**, and **/dev/sdc1**.

1. To use disks in a volume group, label them as LVM physical volumes with the **pvcreate** command.

   **WARNING**

   This command destroys any data on **/dev/sda1**, **/dev/sdb1**, and **/dev/sdc1**.

   ```
   # pvcreate /dev/sda1 /dev/sdb1 /dev/sdc1
     Physical volume "/dev/sda1" successfully created
     Physical volume "/dev/sdb1" successfully created
     Physical volume "/dev/sdc1" successfully created
   ```

2. Create the a volume group that consists of the LVM physical volumes you have created. The following command creates the volume group **myvg**.

   ```
   # vgcreate myvg /dev/sda1 /dev/sdb1 /dev/sdc1
     Volume group "myvg" successfully created
   ```

   You can use the **vgs** command to display the attributes of the new volume group.

   ```
   # vgs
     VG   #PV #LV #SN Attr   VSize  VFree
     myvg   3   0   0 wz--n- 51.45G 51.45G
   ```

3. Create the logical volume from the volume group you have created. The following command creates the logical volume **mylv** from the volume group **myvg**. This example creates a logical volume that uses 2 gigabytes of the volume group.

   ```
   # lvcreate -L 2G -n mylv myvg
     Logical volume "mylv" created
   ```

4. Create a file system on the logical volume. The following command creates an **ext4** file system on the logical volume.

   ```
   # mkfs.ext4 /dev/myvg/mylv
   mke2fs 1.44.3 (10-July-2018)
   Creating filesystem with 524288 4k blocks and 131072 inodes
   ```

```
Filesystem UUID: 616da032-8a48-4cd7-8705-bd94b7a1c8c4
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912

Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done
```

The following commands mount the logical volume and report the file system disk space usage.

```
# mount /dev/myvg/mylv /mnt
# df
Filesystem             1K-blocks     Used  Available Use% Mounted
on
/dev/mapper/myvg-mylv    1998672     6144    1871288   1% /mnt
```

## 2.2. CREATING A RAID0 (STRIPED) LOGICAL VOLUME

A RAID0 logical volume spreads logical volume data across multiple data subvolumes in units of stripe size.

This example procedure creates an LVM RAID0 logical volume called **mylv** that stripes data across the disks at **/dev/sda1**, **/dev/sdb1**, and **/dev/sdc1**.

1. Label the disks you will use in the volume group as LVM physical volumes with the **pvcreate** command.

   **WARNING**

   This command destroys any data on **/dev/sda1**, **/dev/sdb1**, and **/dev/sdc1**.

   ```
   # pvcreate /dev/sda1 /dev/sdb1 /dev/sdc1
     Physical volume "/dev/sda1" successfully created
     Physical volume "/dev/sdb1" successfully created
     Physical volume "/dev/sdc1" successfully created
   ```

2. Create the volume group **myvg**. The following command creates the volume group **myvg**.

   ```
   # vgcreate myvg /dev/sda1 /dev/sdb1 /dev/sdc1
     Volume group "myvg" successfully created
   ```

   You can use the **vgs** command to display the attributes of the new volume group.

   ```
   # vgs
     VG   #PV #LV #SN Attr   VSize  VFree
     myvg   3   0   0 wz--n- 51.45G 51.45G
   ```

3. Create a RAID0 logical volume from the volume group you have created. The following command creates the RAID0 volume **mylv** from the volume group **myvg**. This example creates a logical volume that is 2 gigabytes in size, with three stripes and a stripe size of 4 kilobytes.

```
# lvcreate --type raid0 -L 2G --stripes 3 --stripesize 4 -n mylv
myvg
  Rounding size 2.00 GiB (512 extents) up to stripe boundary size
2.00 GiB(513 extents).
  Logical volume "mylv" created.
```

4. Create a file system on the RAID0 logical volume. The following command creates an **ext4** file system on the logical volume.

```
# mkfs.ext4 /dev/myvg/mylv
mke2fs 1.44.3 (10-July-2018)
Creating filesystem with 525312 4k blocks and 131376 inodes
Filesystem UUID: 9d4c0704-6028-450a-8b0a-8875358c0511
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912

Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done
```

The following commands mount the logical volume and report the file system disk space usage.

```
# mount /dev/myvg/mylv /mnt
# df
Filesystem             1K-blocks     Used  Available Use% Mounted
on
/dev/mapper/myvg-mylv    2002684     6168    1875072   1% /mnt
```

## 2.3. SPLITTING A VOLUME GROUP

In this example procedure, an existing volume group consists of three physical volumes. If there is enough unused space on the physical volumes, a new volume group can be created without adding new disks.

In the initial set up, the logical volume **mylv** is carved from the volume group **myvg**, which in turn consists of the three physical volumes, **/dev/sda1**, **/dev/sdb1**, and **/dev/sdc1**.

After completing this procedure, the volume group **myvg** will consist of **/dev/sda1** and **/dev/sdb1**. A second volume group, **yourvg**, will consist of **/dev/sdc1**.

1. Use the **pvscan** command to determine how much free space is currently available in the volume group.

```
# pvscan
  PV /dev/sda1  VG myvg   lvm2 [17.15 GB / 0    free]
  PV /dev/sdb1  VG myvg   lvm2 [17.15 GB / 12.15 GB free]
  PV /dev/sdc1  VG myvg   lvm2 [17.15 GB / 15.80 GB free]
  Total: 3 [51.45 GB] / in use: 3 [51.45 GB] / in no VG: 0 [0   ]
```

2. Move all the used physical extents in **/dev/sdc1** to **/dev/sdb1** with the **pvmove** command. The **pvmove** command can take a long time to execute.

   ```
   # pvmove /dev/sdc1 /dev/sdb1
     /dev/sdc1: Moved: 14.7%
     /dev/sdc1: Moved: 30.3%
     /dev/sdc1: Moved: 45.7%
     /dev/sdc1: Moved: 61.0%
     /dev/sdc1: Moved: 76.6%
     /dev/sdc1: Moved: 92.2%
     /dev/sdc1: Moved: 100.0%
   ```

   After moving the data, you can see that all of the space on **/dev/sdc1** is free.

   ```
   # pvscan
     PV /dev/sda1   VG myvg   lvm2 [17.15 GB / 0    free]
     PV /dev/sdb1   VG myvg   lvm2 [17.15 GB / 10.80 GB free]
     PV /dev/sdc1   VG myvg   lvm2 [17.15 GB / 17.15 GB free]
     Total: 3 [51.45 GB] / in use: 3 [51.45 GB] / in no VG: 0 [0   ]
   ```

3. To create the new volume group **yourvg**, use the **vgsplit** command to split the volume group **myvg**.
   The following command splits the volume group **yourvg** from the volume group **myvg**, moving the physical volume **/dev/sdc1** into the new volume group **yourvg**.

   ```
   # lvchange -a n /dev/myvg/mylv
   # vgsplit myvg yourvg /dev/sdc1
     Volume group "yourvg" successfully split from "myvg"
   ```

   You can use the **vgs** command to see the attributes of the two volume groups.

   ```
   # vgs
     VG     #PV #LV #SN Attr   VSize  VFree
     myvg     2   1   0 wz--n- 34.30G 10.80G
     yourvg   1   0   0 wz--n- 17.15G 17.15G
   ```

4. After creating the new volume group, create the new logical volume **yourlv**.

   ```
   # lvcreate -L 5G -n yourlv yourvg
     Logical volume "yourlv" created
   ```

5. Create a file system on the new logical volume and mount it.

   ```
   # mkfs.ext4 /dev/yourvg/yourlv
   mke2fs 1.44.3 (10-July-2018)
   Creating filesystem with 524288 4k blocks and 131072 inodes
   Filesystem UUID: 616da032-8a48-4cd7-8705-bd94b7a1c8c4
   Superblock backups stored on blocks:
           32768, 98304, 163840, 229376, 294912

   Allocating group tables: done
   Writing inode tables: done
   Creating journal (16384 blocks): done
   ```

```
Writing superblocks and filesystem accounting information: done

# mount /dev/yourvg/yourlv /mnt
```

## 2.4. REMOVING A DISK FROM A LOGICAL VOLUME

These example procedures show how you can remove a disk from an existing logical volume, either to replace the disk or to use the disk as part of a different volume. In order to remove a disk, you must first move the extents on the LVM physical volume to a different disk or set of disks.

### 2.4.1. Moving Extents to Existing Physical Volumes

In this example, the logical volume is distributed across four physical volumes in the volume group **myvg**.

```
# pvs -o+pv_used
  PV         VG   Fmt  Attr PSize  PFree  Used
  /dev/sda1  myvg lvm2 a-   17.15G 12.15G  5.00G
  /dev/sdb1  myvg lvm2 a-   17.15G 12.15G  5.00G
  /dev/sdc1  myvg lvm2 a-   17.15G 12.15G  5.00G
  /dev/sdd1  myvg lvm2 a-   17.15G  2.15G 15.00G
```

This examples moves the extents off of **/dev/sdb1** so that it can be removed from the volume group.

1. If there are enough free extents on the other physical volumes in the volume group, you can execute the **pvmove** command on the device you want to remove with no other options and the extents will be distributed to the other devices.

   ```
   # pvmove /dev/sdb1
     /dev/sdb1: Moved: 2.0%
    ...
     /dev/sdb1: Moved: 79.2%
    ...
     /dev/sdb1: Moved: 100.0%
   ```

   After the **pvmove** command has finished executing, the distribution of extents is as follows:

   ```
   # pvs -o+pv_used
     PV         VG   Fmt  Attr PSize  PFree  Used
     /dev/sda1  myvg lvm2 a-   17.15G  7.15G 10.00G
     /dev/sdb1  myvg lvm2 a-   17.15G 17.15G     0
     /dev/sdc1  myvg lvm2 a-   17.15G 12.15G  5.00G
     /dev/sdd1  myvg lvm2 a-   17.15G  2.15G 15.00G
   ```

2. Use the **vgreduce** command to remove the physical volume **/dev/sdb1** from the volume group.

   ```
   # vgreduce myvg /dev/sdb1
     Removed "/dev/sdb1" from volume group "myvg"
   # pvs
     PV         VG   Fmt  Attr PSize  PFree
     /dev/sda1  myvg lvm2 a-   17.15G  7.15G
   ```

```
  /dev/sdb1       lvm2 --   17.15G 17.15G
  /dev/sdc1  myvg lvm2 a-   17.15G 12.15G
  /dev/sdd1  myvg lvm2 a-   17.15G  2.15G
```

The disk can now be physically removed or allocated to other users.

## 2.4.2. Moving Extents to a New Disk

In this example, the logical volume is distributed across three physical volumes in the volume group **myvg** as follows:

```
# pvs -o+pv_used
  PV         VG   Fmt  Attr PSize  PFree  Used
  /dev/sda1  myvg lvm2 a-   17.15G  7.15G 10.00G
  /dev/sdb1  myvg lvm2 a-   17.15G 15.15G  2.00G
  /dev/sdc1  myvg lvm2 a-   17.15G 15.15G  2.00G
```

This example procedure moves the extents of **/dev/sdb1** to a new device, **/dev/sdd1**.

1. Create a new physical volume from **/dev/sdd1**.

```
# pvcreate /dev/sdd1
  Physical volume "/dev/sdd1" successfully created
```

2. Add the new physical volume **/dev/sdd1** to the existing volume group **myvg**.

```
# vgextend myvg /dev/sdd1
  Volume group "myvg" successfully extended
# pvs -o+pv_used
  PV         VG   Fmt  Attr PSize  PFree  Used
  /dev/sda1   myvg lvm2 a-   17.15G  7.15G 10.00G
  /dev/sdb1   myvg lvm2 a-   17.15G 15.15G  2.00G
  /dev/sdc1   myvg lvm2 a-   17.15G 15.15G  2.00G
  /dev/sdd1   myvg lvm2 a-   17.15G 17.15G     0
```

3. Use the **pvmove** command to move the data from **/dev/sdb1** to **/dev/sdd1**.

```
# pvmove /dev/sdb1 /dev/sdd1
  /dev/sdb1: Moved: 10.0%
...
  /dev/sdb1: Moved: 79.7%
...
  /dev/sdb1: Moved: 100.0%

# pvs -o+pv_used
  PV          VG   Fmt  Attr PSize  PFree  Used
  /dev/sda1   myvg lvm2 a-   17.15G  7.15G 10.00G
  /dev/sdb1   myvg lvm2 a-   17.15G 17.15G     0
  /dev/sdc1   myvg lvm2 a-   17.15G 15.15G  2.00G
  /dev/sdd1   myvg lvm2 a-   17.15G 15.15G  2.00G
```

4. After you have moved the data off **/dev/sdb1**, you can remove it from the volume group.

```
# vgreduce myvg /dev/sdb1
```

```
Removed "/dev/sdb1" from volume group "myvg"
```

You can now reallocate the disk to another volume group or remove the disk from the system.