Red Hat Enterprise Linux 9

Configuring and managing logical volumes

A guide to the configuration and management of LVM logical volumes

Last Updated: 2022-05-18

Red Hat Enterprise Linux 9 Configuring and managing logical volumes

A guide to the configuration and management of LVM logical volumes

Abstract

This documentation provides instructions on how to manage LVM logical volumes on Red Hat Enterprise Linux 9.

Table of Contents

MAKING OPEN SOURCE MORE INCLUSIVE

Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see our CTO Chris Wright's message .

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

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

- For simple comments on specific passages:
  1. Make sure you are viewing the documentation in the *Multi-page HTML* format. In addition, ensure you see the **Feedback** button in the upper right corner of the document.
  2. Use your mouse cursor to highlight the part of text that you want to comment on.
  3. Click the **Add Feedback** pop-up that appears below the highlighted text.
  4. Follow the displayed instructions.
- For submitting feedback via Bugzilla, create a new 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. OVERVIEW OF LOGICAL VOLUME MANAGEMENT

Logical volume management (LVM) creates a layer of abstraction over physical storage, which helps 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.

1.1. LVM ARCHITECTURE

The following are the components of LVM:

Physical volume

A physical volume (PV) is a partition or whole disk designated for LVM use. For more information, see Managing LVM physical volumes.

Volume group

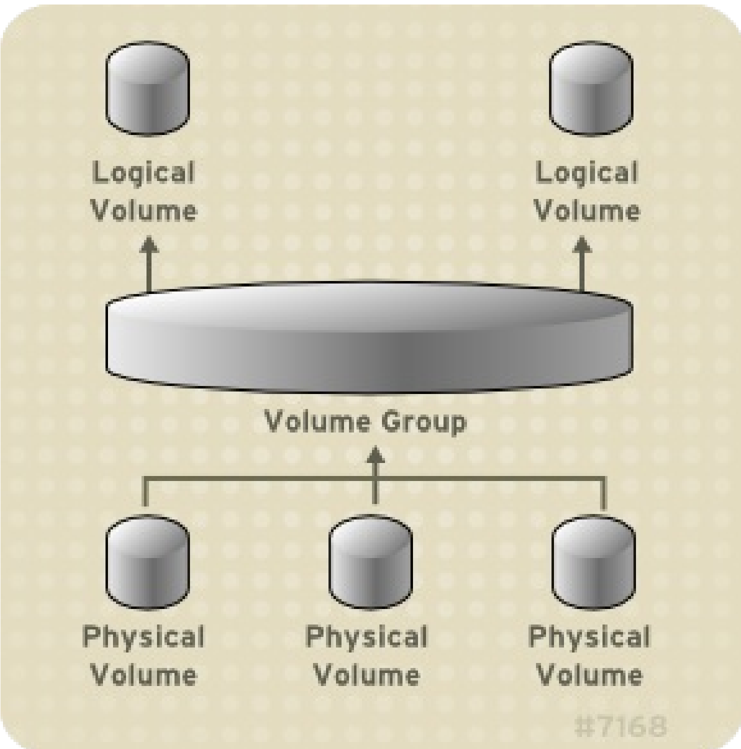
A volume group (VG) is a collection of physical volumes (PVs), which creates a pool of disk space out of which logical volumes can be allocated. For more information, see Managing LVM volume groups.

Logical volume

A logical volume represents a mountable storage device. For more information, see Managing LVM logical volumes.

The following diagram illustrates the components of LVM:

Figure 1.1. LVM logical volume components



1.2. ADVANTAGES OF LVM

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

Flexible capacity

When using logical volumes, you can aggregate devices and partitions into a single logical volume. With this functionality, file systems can extend across multiple devices as though they were a single, large one.

Resizeable storage volumes

You can extend logical volumes or reduce logical volumes in size with simple software commands, without reformatting and repartitioning the underlying 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 with user-defined and custom names.

Striped Volumes

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

RAID volumes

Logical volumes provide a convenient way to configure RAID for your data. This provides protection against device failure and improves performance.

Volume snapshots

You can take snapshots, which is a point-in-time copy of logical volumes 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 physical space.

Cache volumes

A cache logical volume uses a fast block device, such as an SSD drive to improve the performance of a larger and slower block device.

CHAPTER 2. MANAGING LVM PHYSICAL VOLUMES

The physical volume (PV) is a partition or whole disk designated for LVM use. To use the device for an LVM logical volume, the device must be initialized as a physical volume.

If you are using a whole disk device for your physical volume, the disk must have no partition table. For DOS disk partitions, the partition id should be set to 0x8e using the **fdisk** or **cfdisk** command or an equivalent. For whole disk devices only the partition table must be erased, which will effectively destroy all data on that disk. You can remove an existing partition table by zeroing the first sector by using the **dd if=/dev/zero of=*<PhysicalVolume>* bs=512 count=1** command as root

2.1. OVERVIEW OF PHYSICAL VOLUMES

Initializing a block device as a physical volume places a label near the start of the device. The following describes the LVM label:

- An LVM label provides correct identification and device ordering for a physical device. An unlabeled, non-LVM device can change names across reboots depending on the order they are discovered by the system during boot. 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.
- By default, the LVM label is placed in the second 512-byte sector. You can overwrite this default setting 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.

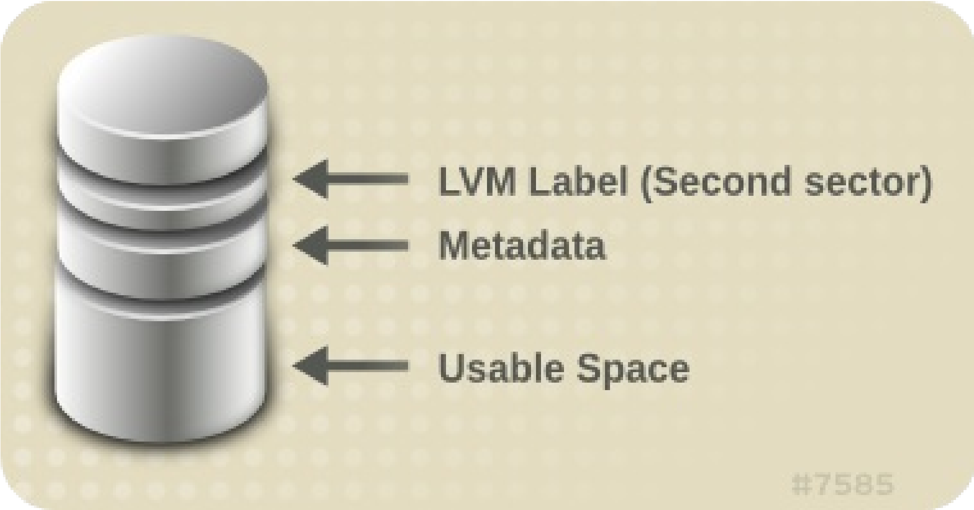
The following describes the LVM metadata:

- 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.

The following diagram illustrates 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 2.1. Physical volume layout



Additional resources

- Multiple partitions on a disk

2.2. MULTIPLE PARTITIONS ON A DISK

You can create physical volumes (PV) out of disk partitions by using LVM.

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.

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.

RAID redundancy

LVM cannot determine that the two physical volumes are on the same device. If you create a RAID logical volume when two physical volumes are on the same device, performance and fault tolerance could be lost.

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 volumes.

Note that although LVM supports using a non-partitioned disk as physical volume, it is recommended to create a single, whole-disk partition because creating a PV without a partition can be problematic in a mixed operating system environment. Other operating systems may interpret the device as free, and overwrite the PV label at the beginning of the drive.

2.3. CREATING LVM PHYSICAL VOLUME

This procedure describes how to create and label LVM physical volumes (PVs).

In this procedure, replace the */dev/vdb1*, */dev/vdb2*, and */dev/vdb3* with the available storage devices in your system.

Prerequisites

- The **lvm2** package is installed.

Procedure

1. Create multiple physical volumes by using the space-delimited device names as arguments to the **pvcreate** command:

```
# pvcreate /dev/vdb1 /dev/vdb2 /dev/vdb3
  Physical volume "/dev/vdb1" successfully created.
  Physical volume "/dev/vdb2" successfully created.
  Physical volume "/dev/vdb3" successfully created.
```

   This places a label on */dev/vdb1*, */dev/vdb2*, and */dev/vdb3*, marking them as physical volumes belonging to LVM.

2. View the created physical volumes by using any one of the following commands as per your requirement:

   a. The **pvdisplay** command, which provides a verbose multi-line output for each physical volume. It displays physical properties, such as size, extents, volume group, and other options in a fixed format:

```
# pvdisplay
--- NEW Physical volume ---
  PV Name               /dev/vdb1
  VG Name
  PV Size               1.00 GiB
[..]
--- NEW Physical volume ---
  PV Name               /dev/vdb2
  VG Name
  PV Size               1.00 GiB
[..]
--- NEW Physical volume ---
  PV Name               /dev/vdb3
  VG Name
  PV Size               1.00 GiB
[..]
```

   b. The **pvs** command provides physical volume information in a configurable form, displaying one line per physical volume:

```
# pvs
  PV         VG  Fmt    Attr    PSize      PFree
/dev/vdb1        lvm2           1020.00m   0
```

```
/dev/vdb2      lvm2        1020.00m   0
/dev/vdb3      lvm2        1020.00m   0
```

c. The **pvscan** command scans all supported LVM block devices in the system for physical volumes. You can define a filter in the **lvm.conf** file so that this command avoids scanning specific physical volumes:

```
# pvscan
  PV  /dev/vdb1              lvm2 [1.00 GiB]
  PV  /dev/vdb2              lvm2 [1.00 GiB]
  PV  /dev/vdb3              lvm2 [1.00 GiB]
```

Additional resources

- **pvcreate(8)**, **pvdisplay(8)**, **pvs(8)**, **pvscan(8)**, and **lvm(8)** man pages

2.4. REMOVING LVM PHYSICAL VOLUMES

If a device is no longer required for use by LVM, you can remove the LVM label by using the **pvremove** command. Executing the **pvremove** command zeroes the LVM metadata on an empty physical volume.

Procedure

1. Remove a physical volume:

```
# pvremove /dev/vdb3
Labels on physical volume "/dev/vdb3" successfully wiped.
```

2. View the existing physical volumes and verify if the required volume is removed:

```
# pvs
  PV         VG   Fmt   Attr   PSize     PFree
/dev/vdb1         lvm2         1020.00m   0
/dev/vdb2         lvm2         1020.00m   0
```

If the physical volume you want to remove is currently part of a volume group, you must remove it from the volume group with the **vgreduce** command. For more information, see Removing physical volumes from a volume group

Additional resources

- **pvremove(8)** man page

CHAPTER 3. MANAGING LVM VOLUME GROUPS

A volume group (VG) is a collection of physical volumes (PVs), which creates a pool of disk space out of which logical volumes (LVs) 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.

3.1. CREATING LVM VOLUME GROUP

This procedure describes how to create an LVM volume group (VG) *myvg*, by using the */dev/vdb1* and */dev/vdb2* physical volumes.

Prerequisites

- The **lvm2** package is installed.
- One or more physical volumes are created. For more information on creating physical volumes, see Creating LVM physical volume.

Procedure

1. Create a volume group:

```
# vgcreate myvg /dev/vdb1 /dev/vdb2
 Volume group "myvg" successfully created.
```

   This creates a VG with the name of *myvg*. The PVs */dev/vdb1* and */dev/vdb2* are the base storage level for the *myvg* VG .

2. View the created volume groups by using any one of the following commands as per your requirement:

   a. The **vgs** command provides volume group information in a configurable form, displaying one line per volume groups:

```
# vgs
  VG   #PV #LV #SN Attr   VSize   VFree
  myvg   4   1   0 wz--n-   3.98g 1008.00m
```

   b. The **vgdisplay** command displays volume group properties such as size, extents, number of physical volumes, and other options in a fixed form. The following example shows the output of the **vgdisplay** command for the volume group *myvg*. If you do not specify a volume group, all existing volume groups are displayed:

```
# vgdisplay myvg _ --- Volume group --- VG Name _myvg
  System ID
  Format                lvm2
  Metadata Areas        4
```

```
  Metadata Sequence No  6
  VG Access             read/write
[..]
```

c. The **vgscan** command scans all supported LVM block devices in the system for volume group:

```
# vgscan
  Found volume group "myvg" using metadata type lvm2
```

3. Optional: Increase a volume group’s capacity by adding one or more free physical volumes:

```
# vgextend myvg /dev/vdb3
Physical volume "/dev/vdb3" successfully created.
Volume group "myvg" successfully extended
```

Additional resources

- **pvcreate(8)**, **vgextend(8)**, **vgdisplay(8)**, **vgs(8)**, **vgscan(8)**, and **lvm(8)** man pages

3.2. COMBINING LVM VOLUME GROUPS

To combine two volume groups into a single volume group, use the **vgmerge** command. You can merge an inactive "source" volume with an active or an inactive "destination" volume if the physical extent sizes of the volume are equal and the physical and logical volume summaries of both volume groups fit into the destination volume groups limits.

Procedure

- Merge the inactive volume group *databases* into the active or inactive volume group *myvg* giving verbose runtime information:

```
# vgmerge -v myvg databases
```

Additional resources

- **vgmerge(8)** man page

3.3. REMOVING PHYSICAL VOLUMES FROM A VOLUME GROUP

To remove unused physical volumes from a volume group, use the **vgreduce** command. The **vgreduce** command shrinks a volume group’s capacity by removing one or more empty physical volumes. This frees those physical volumes to be used in different volume groups or to be removed from the system.

Procedure

1. If the physical volume is still being used, migrate the data to another physical volume from the same volume group :

```
# pvmove /dev/vdb3
  /dev/vdb3: Moved: 2.0%
 ...
```

```
/dev/vdb3: Moved: 79.2%
...
/dev/vdb3: Moved: 100.0%
```

2. If there are no enough free extents on the other physical volumes in the existing volume group:

   a. Create a new physical volume from */dev/vdb4*:

   ```
   # pvcreate /dev/vdb4
     Physical volume "/dev/vdb4" successfully created
   ```

   b. Add the newly created physical volume to the *myvg* volume group:

   ```
   # vgextend myvg /dev/vdb4
     Volume group "myvg" successfully extended
   ```

   c. Move the data from */dev/vdb3* to */dev/vdb4*:

   ```
   # pvmove /dev/vdb3 /dev/vdb4
     /dev/vdb3: Moved: 33.33%
     /dev/vdb3: Moved: 100.00%
   ```

3. Remove the physical volume */dev/vdb3* from the volume group:

   ```
   # vgreduce myvg /dev/vdb3
   Removed "/dev/vdb3" from volume group "myvg"
   ```

Verification

- Verify if the */dev/vdb3* physical volume is removed from the *myvg* volume group:

  ```
  # pvs
    PV         VG   Fmt  Attr PSize    PFree    Used
    /dev/vdb1 myvg  lvm2 a--  1020.00m 0        1020.00m
    /dev/vdb2 myvg  lvm2 a--  1020.00m 0        1020.00m
    /dev/vdb3       lvm2 a--  1020.00m 1008.00m 12.00m
  ```

Additional resources

- **vgreduce(8)**, **pvmove(8)**, and **pvs(8)** man pages

3.4. SPLITTING A LVM VOLUME GROUP

This procedure describes how to split the existing volume group. If there is enough unused space on the physical volumes, a new volume group can be created without adding new disks.

In the initial setup, the volume group *myvg* consists of */dev/vdb1*, */dev/vdb2*, and */dev/vdb3*. After completing this procedure, the volume group *myvg* will consist of */dev/vdb1* and */dev/vdb2*, and the second volume group, *yourvg*, will consist of */dev/vdb3*.

Prerequisites

- You have sufficient space in the volume group. Use the **vgscan** command to determine how much free space is currently available in the volume group.
- Depending on the free capacity in the existing physical volume, move all the used physical extents to other physical volume using the **pvmove** command. For more information, see Removing physical volumes from a volume group .

Procedure

1. Split the existing volume group *myvg* to the new volume group *yourvg*:

```
# vgsplit myvg yourvg /dev/vdb3
  Volume group "yourvg" successfully split from "myvg"
```

> **NOTE**
>
> If you have created a logical volume using the existing volume group, use the following command to deactivate the logical volume:
>
> ```
> # lvchange -a n /dev/myvg/mylv
> ```
>
> For more information on creating logical volumes, see Managing LVM logical volumes.

2. View the attributes of the two volume group:

```
# 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
```

Verification

- Verify if the newly created volume group *yourvg* consists of */dev/vdb3* physical volume:

```
# pvs
  PV          VG     Fmt   Attr   PSize      PFree      Used
  /dev/vdb1 myvg   lvm2   a--    1020.00m    0          1020.00m
  /dev/vdb2 myvg   lvm2   a--    1020.00m    0          1020.00m
  /dev/vdb3 yourvg lvm2   a--    1020.00m   1008.00m    12.00m
```

Additional resources

- **vgsplit(8)**, **vgs(8)**, and **pvs(8)** man pages

3.5. RENAMING LVM VOLUME GROUPS

This procedure renames an existing volume group *myvg* to *myvg1*.

Procedure

1. Deactivate the volume group. If it is a clustered volume group, deactivate the volume group on all nodes where it is active by using the following command on each such node:

```
# vgchange --activate n myvg
```

2. Rename an existing volume group:

```
# vgrename myvg myvg1
Volume group "myvg" successfully renamed to "myvg1"
```

You can also rename the volume group by specifying the full paths to the devices:

```
# vgrename /dev/myvg /dev/myvg1
```

Additional resources

- **vgrename(8)** man page

CHAPTER 4. MANAGING LVM LOGICAL VOLUMES

A logical volume is a virtual, block storage device that a file system, database, or application can use. To create an LVM logical volume, the physical volumes (PVs) are combined into a volume group (VG). This creates a pool of disk space out of which LVM logical volumes (LVs) can be allocated.

4.1. OVERVIEW OF 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. Further, some file systems are not capable of shrinking. 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.

IMPORTANT

On AMD, Intel, ARM systems, and IBM Power Systems servers, the boot loader cannot read LVM volumes. You must make a standard, non-LVM disk partition for your **/boot** partition. On IBM Z, the **zipl** boot loader supports **/boot** on LVM logical volumes with linear mapping. By default, the installation process always creates the / and swap partitions within LVM volumes, with a separate **/boot** partition on a physical volume.

The following are the different types of logical volumes:

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.

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.

RAID logical volumes

LVM supports RAID levels 0, 1, 4, 5, 6, and 10. RAID logical volumes are not cluster-aware. 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.

Thin-provisioned logical volumes (thin volumes)

Using thin-provisioned logical volumes, you can create logical volumes that are larger than the available physical storage. Creating a thinly provisioned set of volumes allows the system to allocate what you use instead of allocating the full amount of storage that is requested

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.

Thin-provisioned snapshot volumes

Using thin-provisioned snapshot volumes, you can have more virtual devices to be stored on the same data volume. Thinly provisioned snapshots are useful because you are not copying all of the data that you are looking to capture at a given time.

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.

4.2. CREATING LVM LOGICAL VOLUME

This procedure describes how to create *mylv* LVM logical volume (LV) from the *myvg* volume group, which is created by using the */dev/vdb1*, */dev/vdb2*, and */dev/vdb3* physical volumes.

Prerequisites

- The **lvm2** package is installed.
- The volume group is created. For more information, see Creating LVM volume group.

Procedure

1. Create a logical volume:

```
# lvcreate -n mylv -L 500M myvg
```

Use the **-n** option to set the LV name to *mylv*, and the **-L** option to set the size of LV in units of Mb, but it is possible to use any other units. The LV type is linear by default, but the user can specify the desired type by using the **--type** option.

IMPORTANT

The command fails if the VG does not have a sufficient number of free physical extents for the requested size and type.

2. View the created logical volumes by using any one of the following commands as per your requirement:

a. The **lvs** command provides logical volume information in a configurable form, displaying one line per logical volume:

```
# lvs
  LV   VG   Attr       LSize   Pool Origin Data%  Meta%  Move Log Cpy%Sync Convert
 mylv myvg -wi-ao----   500.00m
```

b. The **lvdisplay** command displays logical volume properties, such as size, layout, and mapping in a fixed format:

```
# lvdisplay -v /dev/myvg/mylv
```

```
--- Logical volume ---
LV Path                /dev/myvg/mylv
LV Name                mylv
VG Name                myvg
LV UUID                YTnAk6-kMlT-c4pG-HBFZ-Bx7t-ePMk-7YjhaM
LV Write Access        read/write
[..]
```

c. The **lvscan** command scans for all logical volumes in the system and lists them:

```
# lvscan
 ACTIVE            '/dev/myvg/mylv' [500.00 MiB] inherit
```

3. Create a file system on the logical volume. The following command creates an **xfs** file system on the logical volume:

```
# mkfs.xfs /dev/myvg/mylv
meta-data=/dev/myvg/mylv     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.
```

4. Mount the logical volume and report the file system disk space usage:

```
# mount /dev/myvg/mylv /mnt

# df -h
Filesystem             1K-blocks  Used   Available Use% Mounted on

/dev/mapper/myvg-mylv   506528    29388  477140     6%   /mnt
```

Additional resources

- **lvcreate(8)**, **lvdisplay(8)**, **lvs(8)**, **lvscan(8)**, **lvm(8)** and **mkfs.xfs(8)** man pages

4.3. RENAMING LVM LOGICAL VOLUMES

This procedure describes how to rename an existing logical volume *mylv* to *mylv1*.

Procedure

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

```
# umount /mnt
```

Replace */mnt* with the mount point.

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 myvg/mylv
```

3. Rename an existing logical volume:

```
# lvrename myvg mylv mylv1
Logical volume "mylv" successfully renamed to "mylv1"
```

You can also rename the logical volume by specifying the full paths to the devices:

```
# lvrename /dev/myvg/mylv /dev/myvg/mylv1
```

Additional resources

- **lvrename(8)** man page

4.4. REMOVING A DISK FROM A LOGICAL VOLUME

This procedure describes how to 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.

Procedure

1. View the used and free space of physical volumes when using the LV:

```
# pvs -o+pv_used
  PV         VG    Fmt   Attr  PSize     PFree    Used
 /dev/vdb1 myvg  lvm2  a--    1020.00m   0        1020.00m
 /dev/vdb2 myvg  lvm2  a--    1020.00m   0        1020.00m
 /dev/vdb3 myvg  lvm2  a--    1020.00m   1008.00m  12.00m
```

2. Move the data to other physical volume:

   a. If there are enough free extents on the other physical volumes in the existing volume group, use the following command to move the data:

```
# pvmove /dev/vdb3
  /dev/vdb3: Moved: 2.0%
 ...
  /dev/vdb3: Moved: 79.2%
 ...
  /dev/vdb3: Moved: 100.0%
```

   b. If there are no enough free extents on the other physical volumes in the existing volume group, use the following commands to add a new physical volume, extend the volume group using the newly created physical volume, and move the data to this physical volume:

```
# pvcreate /dev/vdb4
```

```
  Physical volume "/dev/vdb4" successfully created

# vgextend myvg /dev/vdb4
  Volume group "myvg" successfully extended

# pvmove /dev/vdb3 /dev/vdb4
  /dev/vdb3: Moved: 33.33%
  /dev/vdb3: Moved: 100.00%
```

3. Remove the physical volume:

```
# vgreduce myvg /dev/vdb3
Removed "/dev/vdb3" from volume group "myvg"
```

If a logical volume contains a physical volume that fails, you cannot use that logical volume. To remove missing physical volumes from a volume group, you can use the **--removemissing** parameter of the **vgreduce** command, if there are no logical volumes that are allocated on the missing physical volumes:

```
# vgreduce --removemissing myvg
```

Additional resources

- **pvmove(8)**, **vgextend(8)**, **vereduce(8)**, and **pvs(8)** man pages

4.5. REMOVING LVM LOGICAL VOLUMES

This procedure describes how to remove an existing logical volume */dev/myvg/mylv1* from the volume group *myvg*.

Procedure

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

```
# umount /mnt
```

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/myvg/mylv1

Do you really want to remove active logical volume "mylv1"? [y/n]: y
Logical volume "mylv1" successfully removed
```

NOTE

In this case, the logical volume has not been deactivated. If you explicitly deactivated the logical volume before removing it, you would not see the prompt verifying whether you want to remove an active logical volume.

Additional resources

- **lvremove(8)** man page

4.6. REMOVING LVM VOLUME GROUPS

This procedure describes how to remove an existing volume group.

Prerequisites

- The volume group contains no logical volumes. To remove logical volumes from a volume group, see Removing LVM logical volumes.

Procedure

1. If the volume group exists in a clustered environment, stop the **lockspace** of the volume group on all other nodes. Use the following command on all nodes except the node where you are performing the removing:

```
# vgchange --lockstop vg-name
```

   Wait for the lock to stop.

2. Remove the volume group:

```
# vgremove vg-name
  Volume group "vg-name" successfully removed
```

Additional resources

- **vgremove(8)** man page

CHAPTER 5. MODIFYING THE SIZE OF A LOGICAL VOLUME

After you have created a logical volume, you can modify the size of the volume.

5.1. GROWING A LOGICAL VOLUME AND FILE SYSTEM

This procedure describes how to extend the logical volume and grow a file system on the same logical volume.

To increase the size of a logical volume, use the **lvextend** command. When you extend the logical volume, you can indicate how much you want to extend the volume, or how large you want it to be after you extend it.

Prerequisites

1. You have an existing logical volume (LV) with a file system on it. Determine the file system type by using the **df -Th** command.
   For more information on creating LV and a file system, see Creating LVM logical volume.

2. You have sufficient space in the volume group to grow your LV and file system. Use the **vgs -o name,vgfree** command to determine the available space.

Procedure

1. Optional: If the volume group has insufficient space to grow your LV, then add a new physical volume to the volume group by using the following command:

   ```
   # vgextend myvg /dev/vdb3
   Physical volume "/dev/vdb3" successfully created.
   Volume group "myvg" successfully extended
   ```

   For more information, see Creating LVM volume group.

2. Now that the volume group is large enough, execute any one of the following steps as per your requirement:

   a. To extend the LV with the provided size, use the following command:

   ```
   # lvextend -L 3G /dev/myvg/mylv
   Size of logical volume myvg/mylv changed from 2.00 GiB (512 extents) to 3.00 GiB (768 extents).
   Logical volume myvg/mylv successfully resized.
   ```

   > **NOTE**
   >
   > You can use the **-r** option of the **lvextend** command to extend the logical volume and resize the underlying file system with a single command:
   >
   > ```
   > # lvextend -r -L 3G /dev/myvg/mylv
   > ```

> **WARNING**
>
> You can also extend the logical volume using the **lvresize** command with the same arguments, but this command does not guarantee against accidental shrinkage.

b. To extend the *mylv* logical volume to fill all of the unallocated space in the *myvg* volume group, use the following command:

```
# lvextend -l +100%FREE /dev/myvg/mylv
 Size of logical volume myvg/mylv changed from 10.00 GiB (2560 extents) to 6.35 TiB (1665465 extents).
 Logical volume myvg/mylv successfully resized.
```

As with the **lvcreate** command, you can use the **-l** argument of the **lvextend** command to specify the number of extents by which to increase the size of the logical volume. You can also use this argument to specify a percentage of the volume group, or a percentage of the remaining free space in the volume group.

3. If you are not using the **r** option with the **lvextend** command to extend the LV and resize the file system with a single command, then resize the file system on the logical volume by using the following command:

```
xfs_growfs /mnt/mnt1/
meta-data=/dev/mapper/myvg-mylv  isize=512    agcount=4, agsize=65536 blks
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=1        finobt=1, sparse=1, rmapbt=0
         =                       reflink=1
data     =                       bsize=4096   blocks=262144, imaxpct=25
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0, ftype=1
log      =internal log           bsize=4096   blocks=2560, version=2
         =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
data blocks changed from 262144 to 524288
```

> **NOTE**
>
> Without the **-D** option, **xfs_growfs** grows the file system to the maximum size supported by the underlying device. For more information, see Increasing the size of an XFS file system.

For resizing an ext4 file system, see Resizing an ext4 file system .

Verification

- Verify if the file system is growing by using the following command:

```
# df -Th
```

```
Filesystem            Type      Size  Used Avail Use% Mounted on
devtmpfs              devtmpfs  1.9G     0  1.9G   0% /dev
tmpfs                 tmpfs     1.9G     0  1.9G   0% /dev/shm
tmpfs                 tmpfs     1.9G  8.6M  1.9G   1% /run
tmpfs                 tmpfs     1.9G     0  1.9G   0% /sys/fs/cgroup
/dev/mapper/rhel-root xfs        45G  3.7G   42G   9% /
/dev/vda1             xfs      1014M  369M  646M  37% /boot
tmpfs                 tmpfs     374M     0  374M   0% /run/user/0
/dev/mapper/myvg-mylv xfs       2.0G   47M  2.0G   3% /mnt/mnt1
```

Additional resources

- **vgextend(8)**, **lvextend(8)**, and **xfs_growfs(8)** man pages

5.2. SHRINKING LOGICAL VOLUMES

You can reduce the size of a logical volume with the **lvreduce** command.

NOTE

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.

WARNING

In most cases, the **lvreduce** command warns about possible data loss and asks for a confirmation. However, you should not rely on these confirmation prompts to prevent data loss because in some cases you will not see these prompts, such as when the logical volume is inactive or the **--resizefs** option is not used.

Note that using the **--test** option of the **lvreduce** command does not indicate where the operation is safe, as this option does not check the file system or test the file system resize.

Procedure

- To shrink the *mylv* logical volume in *myvg* volume group to 64 megabytes, use the following command:

```
# lvreduce --resizefs -L 64M myvg/mylv
fsck from util-linux 2.37.2
/dev/mapper/myvg-mylv: clean, 11/25688 files, 4800/102400 blocks
resize2fs 1.46.2 (28-Feb-2021)
Resizing the filesystem on /dev/mapper/myvg-mylv to 65536 (1k) blocks.
The filesystem on /dev/mapper/myvg-mylv is now 65536 (1k) blocks long.

Size of logical volume myvg/mylv changed from 100.00 MiB (25 extents) to 64.00 MiB (16 extents).
 Logical volume myvg/mylv successfully resized.
```

In this example, *mylv* contains a file system, which this command resizes together with the logical volume.

- Specifying the **-** sign before the resize value indicates that the value will be subtracted from the logical volume's actual size. To shrink a logical volume to an absolute size of 64 megabytes, use the following command:

```
# lvreduce --resizefs -L -64M myvg/mylv
```

Additional resources

- **lvreduce(8)** man page

CHAPTER 6. SNAPSHOT OF LOGICAL VOLUMES

Using the LVM snapshot feature, you can create virtual images of a volume, for example, */dev/sda*, at a particular instant without causing a service interruption.

6.1. OVERVIEW OF SNAPSHOT VOLUMES

When you modify the original volume (the origin) after you take a snapshot, the snapshot feature makes a copy of the modified data area as it was prior to the change so that it can reconstruct the state of the volume. When you create a snapshot, full read and write access to the origin stays possible.

Since 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. It does not provide a substitute for a backup procedure. Snapshot copies are virtual copies and are not an actual media backup.

The size of the snapshot controls the amount of space set aside for storing the changes to the origin volume. For example, if you create a snapshot and then completely overwrite the origin, the snapshot should be at least as big as the origin volume to hold the changes. You should regularly monitor the size of the snapshot. 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 because it contains many writes, such as /**home**.

If a snapshot is full, the snapshot becomes invalid because it can no longer track changes on the origin volume. But you can configure LVM to automatically extend a snapshot whenever its usage exceeds the **snapshot_autoextend_threshold** value to avoid snapshot becoming invalid. Snapshots are fully resizable and you can perform the following operations:

- If you have the storage capacity, you can increase the size of the snapshot volume to prevent it from getting dropped.
- If 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.

The snapshot volume provide the following benefits:

- Most typically, you take a snapshot 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 if the original file system requires file system repair.
- Since the snapshot is read/write, you can test applications against production data by taking a snapshot and running tests against the snapshot without touching the real data.
- You can create LVM volumes for use with Red Hat Virtualization. You can use LVM snapshots 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.

6.2. CREATING A SNAPSHOT OF THE ORIGINAL VOLUME

Use **lvcreate** command with the **-s** or **--size** argument followed by the required size to create a snapshot of the original volume (the origin). A snapshot of a volume is writable. By default, a snapshot volume is activated with the origin during normal activation commands as compared to the thinly-provisioned snapshots. LVM does not support creating a snapshot volume that is larger than the sum of the origin volume's size and the required metadata size for the volume. If you specify a snapshot volume that is larger than this, LVM creates a snapshot volume that is required for the size of the origin.

> **NOTE**
>
> The nodes in a cluster do not support LVM snapshots. You cannot create a snapshot volume in a shared volume group. However, if you need to create a consistent backup of data on a shared logical volume you can activate the volume exclusively and then create the snapshot.

The following procedure creates an origin logical volume named *origin* and a snapshot volume of this original volume named *snap*.

Prerequisites

- You have created volume group *vg001*. For more information, see Creating LVM volume group.

Procedure

1. Create a logical volume named *origin* from the volume group *vg001*:

```
# lvcreate -L 1G -n origin vg001
Logical volume "origin" created.
```

2. Create a snapshot logical volume named *snap* of */dev/vg001/origin* that is *100 MB* in size:

```
# lvcreate --size 100M --name snap --snapshot /dev/vg001/origin
  Logical volume "snap" created.
```

   If the original logical volume contains a file system, you can mount the snapshot logical volume on an arbitrary directory in order to access the contents of the file system to run a backup while the original file system continues to get updated.

3. Display the origin volume and the current percentage of the snapshot volume being used:

```
# lvs -a -o +devices
  LV      VG    Attr       LSize  Pool Origin Data% Meta% Move Log Cpy%Sync Convert
Devices
 origin vg001  owi-a-s---  1.00g                                              /dev/sde1(0)
  snap vg001  swi-a-s--- 100.00m     origin 0.00                                /dev/sde1(256)
```

   You can also display the status of logical volume */dev/vg001/origin* with all the snapshot logical volumes and their status, such as active or inactive by using the **lvdisplay */dev/vg001/origin*** command.

> **WARNING**
>
> Since the snapshot increases in size as the origin volume changes, it is important to monitor the percentage of the snapshot volume regularly with the **lvs** command to be sure it does not become full. A snapshot that is 100% full is lost completely, as a write to unchanged parts of the origin would be unable to succeed without corrupting the snapshot.

4. You can configure LVM to automatically extend a snapshot when its usage exceeds the **snapshot_autoextend_threshold** value to avoid the snapshot becoming invalid when it is 100% full. View the existing values for the **snapshot_autoextend_threshold** and **snapshot_autoextend_percent** options from the **/etc/lvm.conf** file and edit them as per your requirements.
   The following example, sets the **snapshot_autoextend_threshold** option to value less than 100 and **snapshot_autoextend_percent** option to the value depending on your requirement to extend the snapshot volume:

   ```
   # vi /etc/lvm.conf
   snapshot_autoextend_threshold = 70
   snapshot_autoextend_percent = 20
   ```

   You can also extend this snapshot manually by executing the following command:

   ```
   # lvextend -L+100M /dev/vg001/snap
   ```

   > **NOTE**
   >
   > This feature requires unallocated space in the volume group. An automatic extension of a snapshot does not increase the size of a snapshot volume beyond the maximum calculated size that is necessary for the snapshot. Once a snapshot has grown large enough to cover the origin, it is no longer monitored for automatic extension.

Additional resources

- **lvcreate(8)**, **lvextend(8)**, and **lvs(8)** man pages
- **/etc/lvm/lvm.conf** file

6.3. MERGING SNAPSHOT TO ITS ORIGINAL VOLUME

Use the **lvconvert** command with the **--merge** option to merge a snapshot into its original (the origin) volume. You can perform a system rollback if you have lost data or files, or otherwise you have to restore your system to a previous state. After you merge the snapshot volume, the resulting logical volume has the origin volume's name, minor number, and UUID. While the merge is in progress, reads or writes to the origin appear as they were directed to the snapshot being merged. When the merge finishes, the merged snapshot is removed.

If both the origin and snapshot volume are not open and active, the merge starts immediately.

Otherwise, the merge starts after either the origin or snapshot are activated and both are closed. You can merge a snapshot into an origin that cannot be closed, for example a **root** file system, after the origin volume is activated.

Procedure

1. Merge the snapshot volume. The following command merges snapshot volume *vg001/snap* into its *origin*:

```
# lvconvert --merge vg001/snap
Merging of volume vg001/snap started.
  vg001/origin: Merged: 100.00%
```

2. View the origin volume:

```
# lvs -a -o +devices
  LV      VG    Attr       LSize  Pool Origin Data% Meta% Move Log Cpy%Sync Convert
Devices
  origin vg001  owi-a-s---  1.00g                                                     /dev/sde1(0)
```

Additional resources

- **lvconvert(8)** man page

CHAPTER 7. CREATING AND MANAGING THINLY-PROVISIONED VOLUMES (THIN VOLUMES)

Red Hat Enterprise Linux supports thinly-provisioned snapshot volumes and logical volumes.

Logical volumes and snapshot volumes can be thinly provisioned:

- Using thin-provisioned logical volumes, you can create logical volumes that are larger than the available physical storage.
- Using thin-provisioned snapshot volumes, you can have more virtual devices to be stored on the same data volume

7.1. OVERVIEW OF THIN PROVISIONING

Many modern storage stacks now provide the ability to choose between thick provisioning and thin provisioning:

- Thick provisioning provides the traditional behavior of block storage where blocks are allocated regardless of their actual usage.
- Thin provisioning grants the ability to provision a larger pool of block storage that may be larger in size than the physical device storing the data, resulting in over-provisioning. Over-provisioning is possible because individual blocks are not allocated until they are actually used. If you have multiple thin-provisioned devices that share the same pool, then these devices can be over-provisioned.

By using thin provisioning, you can over-commit the physical storage, and instead can manage a pool of free space known as a thin pool. You can allocate this thin pool to an arbitrary number of devices when needed by applications. You can expand the thin pool dynamically when needed for cost-effective allocation of storage space.

For example, if ten users each request a 100GB file system for their application, then you 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.

> **NOTE**
>
> When using thin provisioning, it is important that you monitor the storage pool and add more capacity as the available physical space runs out.

The following are a few advantages of using thin-provisioned devices:

- You can create logical volumes that are larger than the available physical storage.
- You can have more virtual devices to be stored on the same data volume.
- You can create file systems that can grow logically and automatically to support the data requirements and the unused blocks are returned to the pool for use by any file system in the pool

The following are the potential drawbacks of using thin-provisioned devices:

- Thin-provisioned volumes have an inherent risk of running out of available physical storage. If you have over-provisioned your underlying storage, it could possibly result in an outage due to the lack of available physical storage. For example, if you create 10T of thinly provisioned storage with only 1T physical storage for backing, the volumes will become unavailable or unwritable after the 1T is exhausted.

- If volumes are not sending discards to the layers after thin-provisioned devices, then the accounting for usage will not be accurate. For example, placing a file system without the **-o discard mount** option and not running **fstrim** periodically on top of thin-provisioned devices will never unallocate previously used storage. In such cases, you end up using the full provisioned amount over time even if you are not really using it.
- You must monitor the logical and physical usage so as to not run out of available physical space.
- Copy on Write (CoW) operation can be slower on file systems with snapshots.
- Data blocks can be intermixed between multiple file systems leading to random access limitations of the underlying storage even when it does not appear that way to the end user.

7.2. CREATING THINLY-PROVISIONED LOGICAL VOLUMES

Using thin-provisioned logical volumes, you can create logical volumes that are larger than the available physical storage. Creating a thinly provisioned set of volumes allows the system to allocate what you use instead of allocating the full amount of storage that is requested.

Using the **-T** or **--thin** option of the **lvcreate** command, you can create either a thin pool or a thin volume. You can also use the **-T** option of the **lvcreate** command to create both a thin pool and a thin volume at the same time with a single command. This procedure describes how to create and grow thinly-provisioned logical volumes.

Prerequisites

- You have created a volume group. For more information, see Creating LVM volume group.

Procedure

1. Create a thin pool:

```
# lvcreate -L 100M -T vg001/mythinpool
  Thin pool volume with chunk size 64.00 KiB can address at most 15.81 TiB of data.
  Logical volume "mythinpool" created.
```

   Note that since you are creating a pool of physical space, you must specify the size of the pool. The **-T** option of the **lvcreate** command does not take an argument; it determines what type of device is to be created from the other options that are added with the command. You can also create thin pool using additional parameters as shown in the following examples:

   - You can also create a thin pool using the **--thinpool** parameter of the **lvcreate** command. Unlike the **-T** option, the **--thinpool** parameter requires that you specify the name of the thin pool logical volume you are creating. The following example uses the **--thinpool** parameter to create a thin pool named *mythinpool* in the volume group *vg001* that is *100M* in size:

```
# lvcreate -L 100M --thinpool mythinpool vg001
  Thin pool volume with chunk size 64.00 KiB can address at most 15.81 TiB of data.
  Logical volume "mythinpool" created.
```

- As striping is supported for pool creation, you can use the **-i** and **-I** options to create stripes.The following command creates a *100M* thin pool named as *thinpool* in volume group *vg001* with two *64 kB* stripes and a chunk size of *256 kB*. It also creates a *1T* thin volume named *vg001/thinvolume*.

  NOTE

  Ensure that there are two physical volumes with sufficient free space in the volume group or you cannot create the thin pool.

  ```
  # lvcreate -i 2 -I 64 -c 256 -L 100M -T vg001/thinpool -V 1T --name thinvolume
  ```

2. Create a thin volume:

```
# lvcreate -V 1G -T vg001/mythinpool -n thinvolume
  WARNING: Sum of all thin volume sizes (1.00 GiB) exceeds the size of thin pool
vg001/mythinpool (100.00 MiB).
  WARNING: You have not turned on protection against thin pools running out of space.
  WARNING: Set activation/thin_pool_autoextend_threshold below 100 to trigger automatic
extension of thin pools before they get full.
  Logical volume "thinvolume" created.
```

In this case, you are specifying virtual size for the volume that is greater than the pool that contains it. You can also create thin volumes using additional parameters as shown in the following examples:

- To create both a thin volume and a thin pool, use the **-T** option of the **lvcreate** command and specify both the size and virtual size argument:

  ```
  # lvcreate -L 100M -T vg001/mythinpool -V 1G -n thinvolume
    Thin pool volume with chunk size 64.00 KiB can address at most 15.81 TiB of data.
    WARNING: Sum of all thin volume sizes (1.00 GiB) exceeds the size of thin pool
  vg001/mythinpool (100.00 MiB).
    WARNING: You have not turned on protection against thin pools running out of space.
    WARNING: Set activation/thin_pool_autoextend_threshold below 100 to trigger
  automatic extension of thin pools before they get full.
    Logical volume "thinvolume" created.
  ```

- To use the remaining free space to create a thin volume and thin pool, use the **100%FREE** option:

  ```
  # lvcreate -V 1G -l 100%FREE -T vg001/mythinpool -n thinvolume
  Thin pool volume with chunk size 64.00 KiB can address at most <15.88 TiB of data.
    Logical volume "thinvolume" created.
  ```

- To convert an existing logical volume to a thin pool volume, use the **--thinpool** parameter of the **lvconvert** command. You must also use the **--poolmetadata** parameter in conjunction with the **--thinpool** parameter to convert an existing logical volume to a thin pool volume's metadata volume.
  The following example converts the existing logical volume *lv1* in volume group *vg001* to a thin pool volume and converts the existing logical volume *lv2* in volume group *vg001* to the metadata volume for that thin pool volume:

```
# lvconvert --thinpool vg001/lv1 --poolmetadata vg001/lv2
  Converted vg001/lv1 to thin pool.
```

NOTE

Converting a logical volume to a thin pool volume or a thin pool metadata volume destroys the content of the logical volume, as **lvconvert** does not preserve the content of the devices but instead overwrites the content.

- By default, the **lvcreate** command sets the size of the thin pool's metadata logical volume according to the following formula:

```
Pool_LV_size / Pool_LV_chunk_size * 64
```

If you have large numbers of snapshots or if you have have small chunk sizes for your thin pool and thus expect significant growth of the size of the thin pool at a later time, you may need to increase the default value of the thin pool's metadata volume using the **--poolmetadatasize** parameter of the **lvcreate** command. The supported value for the thin pool's metadata logical volume is in the range between 2MiB and 16GiB.

The following example illustrates how to increase the default value of the thin pools' metadata volume:

```
# lvcreate -V 1G -l 100%FREE -T vg001/mythinpool --poolmetadatasize 16M -n
thinvolume
Thin pool volume with chunk size 64.00 KiB can address at most 15.81 TiB of data.
  Logical volume "thinvolume" created.
```

3. View the created thin pool and thin volume:

```
# lvs -a -o +devices
  LV                VG    Attr       LSize   Pool       Origin Data%  Meta%  Move Log Cpy%Sync
Convert Devices
  [lvol0_pmspare]    vg001 ewi-------   4.00m                                              /dev/sda(0)
  mythinpool         vg001 twi-aotz-- 100.00m                 0.00   10.94
mythinpool_tdata(0)
  [mythinpool_tdata] vg001 Twi-ao---- 100.00m
/dev/sda(1)
  [mythinpool_tmeta] vg001 ewi-ao----   4.00m
/dev/sda(26)
  thinvolume         vg001 Vwi-a-tz--   1.00g mythinpool        0.00
```

4. Optional: Extend the size of a thin pool with the **lvextend** command. You cannot, however, reduce the size of a thin pool.

NOTE

This command fails if you use **-l 100%FREE** argument while creating a thin pool and thin volume.

The following command resizes an existing thin pool that is *100M* in size by extending it another *100M*:

```
# lvextend -L+100M vg001/mythinpool
  Size of logical volume vg001/mythinpool_tdata changed from 100.00 MiB (25 extents) to
200.00 MiB (50 extents).
  WARNING: Sum of all thin volume sizes (1.00 GiB) exceeds the size of thin pool
vg001/mythinpool (200.00 MiB).
  WARNING: You have not turned on protection against thin pools running out of space.
  WARNING: Set activation/thin_pool_autoextend_threshold below 100 to trigger automatic
extension of thin pools before they get full.

  Logical volume vg001/mythinpool successfully resized
```

```
# lvs -a -o +devices
  LV                VG    Attr       LSize   Pool       Origin Data%  Meta%  Move Log Cpy%Sync
Convert Devices
  [lvol0_pmspare]    vg001 ewi-------   4.00m                                         /dev/sda(0)
  mythinpool         vg001 twi-aotz-- 200.00m                0.00   10.94
mythinpool_tdata(0)
  [mythinpool_tdata] vg001 Twi-ao---- 200.00m
/dev/sda(1)
  [mythinpool_tdata] vg001 Twi-ao---- 200.00m
/dev/sda(27)
  [mythinpool_tmeta] vg001 ewi-ao----   4.00m
/dev/sda(26)
  thinvolume         vg001 Vwi-a-tz--   1.00g mythinpool        0.00
```

5. Optional: To rename the thin pool and thin volume, use the following command:

```
# lvrename vg001/mythinpool vg001/mythinpool1
  Renamed "mythinpool" to "mythinpool1" in volume group "vg001"

# lvrename vg001/thinvolume vg001/thinvolume1
  Renamed "thinvolume" to "thinvolume1" in volume group "vg001"
```

View the thin pool and thin volume after renaming:

```
# lvs
  LV          VG       Attr     LSize   Pool       Origin Data%  Move Log Copy%  Convert
mythinpool1 vg001   twi-a-tz 100.00m                   0.00
thinvolume1 vg001   Vwi-a-tz   1.00g mythinpool1        0.00
```

6. Optional: To remove the thin pool, use the following command:

```
# lvremove -f vg001/mythinpool1
  Logical volume "thinvolume1" successfully removed.
  Logical volume "mythinpool1" successfully removed.
```

Additional resources

- **lvcreate(8)**, **lvrename(8)**, **lvs(8)**, and **lvconvert(8)** man pages

7.3. THINLY-PROVISIONED SNAPSHOT VOLUMES

Red Hat Enterprise Linux supports thinly-provisioned snapshot volumes. A snapshot of a thin logical volume also creates a thin logical volume (LV). A thin snapshot volume has the same characteristics as any other thin volume. You can independently activate the volume, extend the volume, rename the volume, remove the volume, and even snapshot the volume.

> **NOTE**
>
> Similarly to all LVM snapshot volumes, and 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.

Traditional snapshots must allocate new space for each snapshot created, where data is preserved as changes are made to the origin. But thin-provisioning snapshots share the same space with the origin. Snapshots of thin LVs are efficient because the data blocks common to a thin LV and any of its snapshots are shared. You can create snapshots of thin LVs or from the other thin snapshots. Blocks common to recursive snapshots are also shared in the thin pool.

Thin snapshot volumes provide the following benefits:

- Increasing the number of snapshots of the origin has a negligible impact on performance.
- A thin snapshot volume can reduce disk usage because only the new data is written and is not copied to each snapshot.
- There is no need to simultaneously activate the thin snapshot volume with the origin, which is a requirement of traditional snapshots.
- When restoring an origin from a snapshot, it is not required to merge the thin snapshot. You can remove the origin and instead use the snapshot. Traditional snapshots have a separate volume where they store changes that must be copied back, that is, merged to the origin to reset it.
- There is a significantly higher limit on the number of allowed snapshots as compared to the traditional snapshots.

Although there are many advantages for using thin snapshot volumes, there are some use cases for which the traditional LVM snapshot volume feature might be more appropriate to your needs. You can use traditional snapshots with all types of volumes. However, to use thin-snapshots requires you to use thin-provisioning.

> **NOTE**
>
> You cannot limit the size of a thin snapshot volume; the snapshot uses all of the space in the thin pool, if necessary. In general, you should consider the specific requirements of your site when deciding which snapshot format to use.

By default, a thin snapshot volume is skipped during normal activation commands.

7.4. CREATING THINLY-PROVISIONED SNAPSHOT VOLUMES

Using thin-provisioned snapshot volumes, you can have more virtual devices stored on the same data volume.

IMPORTANT

When creating a thin snapshot volume, do not specify the size of the volume. If you specify a size parameter, the snapshot that will be created will not be a thin snapshot volume and will not use the thin pool for storing data. For example, the command **lvcreate -s vg/thinvolume -L10M** will not create a thin snapshot, even though the origin volume is a thin volume.

Thin snapshots can be created for thinly-provisioned origin volumes, or for origin volumes that are not thinly-provisioned. The following procedure describes different ways to create a thinly-provisioned snapshot volume.

Prerequisites

- You have created a thinly-provisioned logical volume. For more information, see Creating thinly-provisioned logical volumes.

Procedure

- Create a thinly-provisioned snapshot volume. The following command creates a thinly-provisioned snapshot volume named as *mysnapshot1* of the thinly-provisioned logical volume *vg001/thinvolume*:

```
# lvcreate -s --name mysnapshot1 vg001/thinvolume
  Logical volume "mysnapshot1" created
```

```
# lvs
  LV          VG       Attr     LSize   Pool       Origin     Data%  Move Log Copy%  Convert
  mysnapshot1 vg001    Vwi-a-tz   1.00g mythinpool thinvolume   0.00
  mythinpool  vg001    twi-a-tz 100.00m                         0.00
  thinvolume  vg001    Vwi-a-tz   1.00g mythinpool              0.00
```

NOTE

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. For information on extending the size of a thin volume, see Creating thinly-provisioned logical volumes .

- You can also create a thinly-provisioned snapshot of a non-thinly-provisioned logical volume. Since the non-thinly-provisioned logical volume is not contained within a thin pool, it is referred to as an external origin. External origin volumes can be used and shared by many thinly-provisioned snapshot volumes, even from different thin pools. The external origin must be inactive and read-only at the time the thinly-provisioned snapshot is created.
  The following example creates a thin snapshot volume of the read-only, inactive logical volume named *origin_volume*. The thin snapshot volume is named *mythinsnap*. The logical volume *origin_volume* then becomes the thin external origin for the thin snapshot volume *mythinsnap* in volume group *vg001* that uses the existing thin pool *vg001/pool*. The origin volume must be in the same volume group as the snapshot volume. Do not specify the volume group when specifying the origin logical volume.

```
# lvcreate -s --thinpool vg001/pool origin_volume --name mythinsnap
```

- You can create a second thinly-provisioned snapshot volume of the first snapshot volume by executing the following command.

```
# lvcreate -s vg001/mysnapshot1 --name mysnapshot2
Logical volume "mysnapshot2" created.
```

  To create a third thinly-provisioned snapshot volume, use the following command:

```
# lvcreate -s vg001/mysnapshot2 --name mysnapshot3
Logical volume "mysnapshot3" created.
```

Verification

- Display a list of all ancestors and descendants of a thin snapshot logical volume:

```
$ lvs -o name,lv_ancestors,lv_descendants vg001
  LV          Ancestors                          Descendants
  mysnapshot2 mysnapshot1,thinvolume             mysnapshot3
  mysnapshot1 thinvolume              mysnapshot2,mysnapshot3
  mysnapshot3 mysnapshot2,mysnapshot1,thinvolume
  mythinpool
  thinvolume                          mysnapshot1,mysnapshot2,mysnapshot3
```

  Here,

- *thinvolume* is an origin volume in volume group *vg001*.
- *mysnapshot1* is a snapshot of *thinvolume*
- *mysnapshot2* is a snapshot of *mysnapshot1*
- *mysnapshot3* is a snapshot of *mysnapshot2*

> **NOTE**
>
> The **lv_ancestors** and **lv_descendants** fields display existing dependencies. However, they do not track removed entries which can break a dependency chain if the entry was removed from the middle of the chain.

Additional resources

- **lvcreate(8)** man page

CHAPTER 8. TROUBLESHOOTING LVM

You can use LVM tools to troubleshoot a variety of issues in LVM volumes and groups.

8.1. GATHERING DIAGNOSTIC DATA ON LVM

If an LVM command is not working as expected, you can gather diagnostics in the following ways.

Procedure

- Use the following methods to gather different kinds of diagnostic data:
    - Add the **-v** argument to any LVM command to increase the verbosity level of the command output. Verbosity can be further increased by adding additional **v's**. A maximum of four such **v's** is allowed, for example, **-vvvv**.
    - In the **log** section of the **/etc/lvm/lvm.conf** configuration file, increase the value of the **level** option. This causes LVM to provide more details in the system log.
    - If the problem is related to the logical volume activation, enable LVM to log messages during the activation:
        i. Set the **activation = 1** option in the **log** section of the **/etc/lvm/lvm.conf** configuration file.
        ii. Execute the LVM command with the **-vvvv** option.
        iii. Examine the command output.
        iv. Reset the **activation** option to **0**.
        If you do not reset the option to **0**, the system might become unresponsive during low memory situations.
    - Display an information dump for diagnostic purposes:

      ```
      # lvmdump
      ```
    - Display additional system information:

      ```
      # lvs -v
      ```

      ```
      # pvs --all
      ```

      ```
      # dmsetup info --columns
      ```
    - Examine the last backup of the LVM metadata in the **/etc/lvm/backup/** directory and archived versions in the **/etc/lvm/archive/** directory.
    - Check the current configuration information:

      ```
      # lvmconfig
      ```
    - Check the **/run/lvm/hints** cache file for a record of which devices have physical volumes on them.

Additional resources

- **lvmdump(8)** man page

8.2. DISPLAYING INFORMATION ON FAILED LVM DEVICES

You can display information about a failed LVM volume that can help you determine why the volume failed.

Procedure

- Display the failed volumes using the **vgs** or **lvs** utility.

Example 8.1. Failed volume groups

In this example, one of the devices that made up the volume group *myvg* failed. The volume group is unusable but you can see information about the failed device.

```
# vgs --options +devices
 /dev/vdb1: open failed: No such device or address
 /dev/vdb1: open failed: No such device or address
  WARNING: Couldn't find device with uuid 42B7bu-YCMp-CEVD-CmKH-2rk6-fiO9-
z1lf4s.
  WARNING: VG myvg is missing PV 42B7bu-YCMp-CEVD-CmKH-2rk6-fiO9-z1lf4s (last
written to /dev/sdb1).
  WARNING: Couldn't find all devices for LV myvg/mylv while checking used and assumed
devices.

VG    #PV #LV #SN Attr   VSize  VFree  Devices
myvg   2   2   0 wz-pn- <3.64t <3.60t [unknown](0)
myvg   2   2   0 wz-pn- <3.64t <3.60t [unknown](5120),/dev/vdb1(0)
```

Example 8.2. Failed logical volume

In this example, one of the devices failed due to which the logical volume in the volume group failed. The command output shows the failed logical volumes.

```
# lvs --all --options +devices

  /dev/vdb1: open failed: No such device or address
  /dev/vdb1: open failed: No such device or address
  WARNING: Couldn't find device with uuid 42B7bu-YCMp-CEVD-CmKH-2rk6-fiO9-
z1lf4s.
  WARNING: VG myvg is missing PV 42B7bu-YCMp-CEVD-CmKH-2rk6-fiO9-z1lf4s (last
written to /dev/sdb1).
  WARNING: Couldn't find all devices for LV myvg/mylv while checking used and assumed
devices.

  LV    VG  Attr       LSize  Pool Origin Data%  Meta%  Move Log Cpy%Sync Convert
Devices
  mylv myvg -wi-a---p- 20.00g                                                  [unknown](0)
[unknown](5120),/dev/sdc1(0)
```

8.3. REMOVING LOST LVM PHYSICAL VOLUMES FROM A VOLUME GROUP

If a physical volume fails, you can activate the remaining physical volumes in the volume group and remove all the logical volumes that used that physical volume from the volume group.

Procedure

1. Activate the remaining physical volumes in the volume group:

   ```
   # vgchange --activate y --partial myvg
   ```

2. Check which logical volumes will be removed:

   ```
   # vgreduce --removemissing --test myvg
   ```

3. Remove all the logical volumes that used the lost physical volume from the volume group:

   ```
   # vgreduce --removemissing --force myvg
   ```

4. Optional: If you accidentally removed logical volumes that you wanted to keep, you can reverse the **vgreduce** operation:

   ```
   # vgcfgrestore myvg
   ```

   WARNING

   If you remove a thin pool, LVM cannot reverse the operation.

8.4. FINDING THE METADATA OF A MISSING LVM PHYSICAL VOLUME

If the volume group's metadata area of a physical volume is accidentally overwritten or otherwise destroyed, you get an error message indicating that the metadata area is incorrect, or that the system was unable to find a physical volume with a particular UUID.

This procedure finds the latest archived metadata of a physical volume that is missing or corrupted.

Procedure

1. Find the archived metadata file of the volume group that contains the physical volume. The archived metadata files are located at the **/etc/lvm/archive/*volume-group-name_backup-number*.vg** path:

   ```
   # cat /etc/lvm/archive/myvg_00000-1248998876.vg
   ```

   Replace *00000-1248998876* with the backup-number. Select the last known valid metadata file, which has the highest number for the volume group.

2. Find the UUID of the physical volume. Use one of the following methods.

   - List the logical volumes:

```
# lvs --all --options +devices

  Couldn't find device with uuid 'FmGRh3-zhok-iVI8-7qTD-S5BI-MAEN-NYM5Sk'.
```

   - Examine the archived metadata file. Find the UUID as the value labeled **id =** in the **physical_volumes** section of the volume group configuration.

   - Deactivate the volume group using the **--partial** option:

```
# vgchange --activate n --partial myvg

  PARTIAL MODE. Incomplete logical volumes will be processed.
  WARNING: Couldn't find device with uuid 42B7bu-YCMp-CEVD-CmKH-2rk6-fiO9-z1lf4s.
  WARNING: VG myvg is missing PV 42B7bu-YCMp-CEVD-CmKH-2rk6-fiO9-z1lf4s (last written to /dev/vdb1).
  0 logical volume(s) in volume group "myvg" now active
```

8.5. RESTORING METADATA ON AN LVM PHYSICAL VOLUME

This procedure restores metadata on a physical volume that is either corrupted or replaced with a new device. You might be able to recover the data from the physical volume by rewriting the metadata area on the physical volume.

> **WARNING**
>
> Do not attempt this procedure on a working LVM logical volume. You will lose your data if you specify the incorrect UUID.

Prerequisites

- You have identified the metadata of the missing physical volume. For details, see Finding the metadata of a missing LVM physical volume.

Procedure

1. Restore the metadata on the physical volume:

```
# pvcreate --uuid physical-volume-uuid \
          --restorefile /etc/lvm/archive/volume-group-name_backup-number.vg \
          block-device
```

NOTE

The command overwrites only the LVM metadata areas and does not affect the existing data areas.

Example 8.3. Restoring a physical volume on */dev/vdb1*

The following example labels the ***/dev/vdb1*** device as a physical volume with the following properties:

- The UUID of **FmGRh3-zhok-iVI8-7qTD-S5BI-MAEN-NYM5Sk**
- The metadata information contained in **VG_00050.vg**, which is the most recent good archived metadata for the volume group

```
# pvcreate --uuid "FmGRh3-zhok-iVI8-7qTD-S5BI-MAEN-NYM5Sk" \
          --restorefile /etc/lvm/archive/VG_00050.vg \
          /dev/vdb1

  ...
  Physical volume "/dev/vdb1" successfully created
```

2. Restore the metadata of the volume group:

```
# vgcfgrestore myvg

  Restored volume group myvg
```

3. Display the logical volumes on the volume group:

```
# lvs --all --options +devices myvg
```

The logical volumes are currently inactive. For example:

```
  LV     VG    Attr   LSize   Origin Snap%  Move Log Copy%  Devices
  mylv myvg   -wi--- 300.00G                               /dev/vdb1 (0),/dev/vdb1(0)
  mylv myvg   -wi--- 300.00G                               /dev/vdb1 (34728),/dev/vdb1(0)
```

4. If the segment type of the logical volumes is RAID, resynchronize the logical volumes:

```
# lvchange --resync myvg/mylv
```

5. Activate the logical volumes:

```
# lvchange --activate y myvg/mylv
```

6. If the on-disk LVM metadata takes at least as much space as what overrode it, this procedure can recover the physical volume. If what overrode the metadata went past the metadata area, the data on the volume may have been affected. You might be able to use the **fsck** command to recover that data.

Verification steps

- Display the active logical volumes:

```
# lvs --all --options +devices

 LV     VG   Attr   LSize   Origin Snap%  Move Log Copy%  Devices
 mylv myvg   -wi--- 300.00G                               /dev/vdb1 (0),/dev/vdb1(0)
 mylv myvg   -wi--- 300.00G                               /dev/vdb1 (34728),/dev/vdb1(0)
```

8.6. ROUNDING ERRORS IN LVM OUTPUT

LVM commands that report the space usage in volume groups round the reported number to **2** decimal places to provide human-readable output. This includes the **vgdisplay** and **vgs** utilities.

As a result of the rounding, the reported value of free space might be larger than what the physical extents on the volume group provide. If you attempt to create a logical volume the size of the reported free space, you might get the following error:

```
Insufficient free extents
```

To work around the error, you must examine the number of free physical extents on the volume group, which is the accurate value of free space. You can then use the number of extents to create the logical volume successfully.

8.7. PREVENTING THE ROUNDING ERROR WHEN CREATING AN LVM VOLUME

When creating an LVM logical volume, you can specify the number of logical extents of the logical volume to avoid rounding error.

Procedure

1. Find the number of free physical extents in the volume group:

```
# vgdisplay myvg
```

Example 8.4. Free extents in a volume group

For example, the following volume group has 8780 free physical extents:

```
--- Volume group ---
 VG Name               myvg
 System ID
 Format                lvm2
 Metadata Areas        4
 Metadata Sequence No  6
 VG Access             read/write
[...]
Free  PE / Size       8780 / 34.30 GB
```

2. Create the logical volume. Enter the volume size in extents rather than bytes.

Example 8.5. Creating a logical volume by specifying the number of extents

```
# lvcreate --extents 8780 --name mylv myvg
```

Example 8.6. Creating a logical volume to occupy all the remaining space

Alternatively, you can extend the logical volume to use a percentage of the remaining free space in the volume group. For example:

```
# lvcreate --extents 100%FREE --name mylv myvg
```

Verification steps

- Check the number of extents that the volume group now uses:

```
# vgs --options +vg_free_count,vg_extent_count

  VG     #PV #LV #SN  Attr   VSize   VFree  Free  #Ext
  myvg   2   1   0   wz--n- 34.30G    0    0     8780
```