Oracle Linux 7

Managing Storage and Storage Devices

F32385-05
October 2022

Oracle Linux 7 Managing Storage and Storage Devices,

F32385-05

Contents

Preface

Oracle® Linux 7: Managing Storage and Storage Devices describes how to configure and manage disk partitions, swap space, logical volumes, software RAID (redundant array of independent disks), block device encryption, iSCSI storage, and multipathing.

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at https://www.oracle.com/corporate/accessibility/.

For information about the accessibility of the Oracle Help Center, see the Oracle Accessibility Conformance Report at https://www.oracle.com/corporate/accessibility/templates/t2-11535.html.

Access to Oracle Support for Accessibility

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit https://www.oracle.com/corporate/accessibility/learning-support.html#support-tab.

Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry

standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

1

Configuring Storage Devices

This chapter describes how to configure and manage disk partitions and swap spaces.

For additional information about Oracle Linux products featuring storage and storage management, refer to the following guides:

- Oracle Linux: Gluster Storage for Oracle Linux User's Guide
- Oracle Linux: Ceph Storage User's Guide

Working With Disk Partitions

Partitioning a disk drive involves dividing it into one or more reserved areas, called *partitions*, and then stores information about these partitions in the partition table on the disk. The operating system treats each partition as a separate disk that can contain a file system.

Oracle Linux requires one partition for the `root` file system. Typically, two other partitions are used for swap space and the boot file system. On x86 and x86_64 systems, the system BIOS can usually access only the first 1024 cylinders of the disk at boot time. Configuring a separate boot partition in this region on the disk enables the GRand Unified Bootloader (GRUB) to access the kernel image and any other files that are required to boot the system.

You can create additional partitions to do the following: simplify backups, enhance system security, as well as meet other needs, such as setting up development sandboxes and test areas. Data that frequently changes, such as user home directories, databases, and log file directories, is typically assigned to separate partitions to facilitate backups.

The partitioning scheme for hard disks with a master boot record (MBR) enables you to create up to four *primary partitions*. If you need more than four partitions, you can divide one of the primary partitions into up to 11 *logical partitions*. The primary partition that contains the logical partitions is known as an *extended partition*. The Master Boot Record (MBR) scheme supports disks up to 2 TB in size.

On hard disks with a GUID Partition Table (GPT), you can configure up to 128 partitions and there is no concept of extended or logical partitions. You should configure a GPT if the disk is larger than 2 TB.

You can create and manage MBRs by using the `fdisk` command. If you want to create a GPT, use the `parted` command.

> **Note:**
>
> When partitioning a block storage device, align primary and logical partitions on one-megabyte (1048576 bytes) boundaries. If partitions, file system blocks, or RAID stripes are incorrectly aligned and overlap the boundaries of the underlying storage's sectors or pages, the device controller has to modify twice as many sectors or pages than if correct alignment is used. This recommendation applies to most block storage devices, including hard disk drives (*spinning rust*), solid state drives (SSDs), logical unit numbers (LUNs) on storage arrays, and host RAID adapters.

Managing Partition Tables With the fdisk Utility

> **Caution:**
>
> If any partition on the disk to be configured by using `fdisk` is currently mounted, unmount it before running the `fdisk` utility on the disk. Similarly, if any partition is being used as swap space, use the `swapoff` command to disable the partition.
>
> Before running the `fdisk` utility on a disk that contains data, first back up the data onto another disk or medium.
>
> You cannot use the `fdisk` utility to manage a GPT hard disk.

Use the `fdisk` utility to do the following:

- Create a partition table.
- View an existing partition table.
- Add and delete partitions.

Alternatively, you can use the `cfdisk` utility, which is a text-based, graphical version of the `fdisk` utility.

You can use the `fdisk` utility interactively; or, you can use command-line options and arguments to specify partitions. When you run `fdisk` interactively, you specify only the name of the disk device as an argument, for example:

```
sudo fdisk /dev/sda

WARNING: DOS-compatible mode is deprecated. It's strongly recommended to
         switch off the mode (command 'c') and change display units to
         sectors (command 'u').

Command (m for help):
```

Based on the strong recommendation in the output message, type `c` at the `Command` prompt to switch off DOS-compatibility mode, then `u` to use sectors, and the `p` command to display the partition table:

```
Command (m for help): c
DOS Compatibility flag is not set

Command (m for help): u
Changing display/entry units to sectors

Command (m for help): p

Disk /dev/sda: 42.9 GB, 42949672960 bytes
255 heads, 63 sectors/track, 5221 cylinders, total 83886080 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x0002a95d

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *        2048     1026047      512000   83  Linux
/dev/sda2         1026048    83886079    41430016   8e  Linux LVM
```

The previous example output shows that `/dev/sda` is a 42.9 GB disk. As modern hard disks support logical block addressing (LBA), any information about the numbers of heads and sectors per track is irrelevant and most likely fictitious. The start and end offsets of each partition from the beginning of the disk are shown in units of sectors. The partition table is displayed after the device summary and displays the following information:

Device
The device that corresponds to the partition.

Boot
Specifies `*` if the partition contains the files that the GRUB bootloader needs to boot the system. Only one partition can be bootable.

Start and End
The start and end offsets in sectors. All partitions are aligned on one-megabyte boundaries.

Blocks
The size of the partition in one-kilobyte blocks.

Id and System
The partition type. The following partition types are typically used with Oracle Linux:

5 Extended
An extended partition that can contain up to four logical partitions.

82 Linux swap
Swap space partition.

83 Linux
Linux partition for a file system that is not managed by LVM. This is the default partition type.

8e Linux LVM
Linux partition that is managed by LVM.

The `n` command creates a new partition. For example, to create partition table entries for two Linux partitions on `/dev/sdc`, one of which is 5 GB in size and the other occupies the remainder of the disk:

```
sudo fdisk -cu /dev/sdc

...
Command (m for help): n
Command action
    e    extended
    p    primary partition (1-4)
p
Partition number (1-4): 1
First sector (2048-25165823, default 2048):  2048
Last sector, +sectors or +size{K,M,G} (2048-25165823, default 25165823): +5G

Command (m for help): n
Command action
    e    extended
    p    primary partition (1-4)
p
Partition number (1-4): 2
First sector (10487808-25165823, default 10487808): <Enter>
Using default value 10487808
Last sector, +sectors or +size{K,M,G} (10487808-25165823, default 25165823):
<Enter>
Using default value 25165823

Command (m for help): p

Disk /dev/sdc: 12.9 GB, 12884901888 bytes
255 heads, 63 sectors/track, 1566 cylinders, total 25165824 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0xe6d3c9f6

   Device Boot      Start         End      Blocks   Id  System
/dev/sdc1            2048    10487807     5242880   83  Linux
/dev/sdc2        10487808    25165823     7339008   83  Linux
```

The `t` command enables you to change the type of a partition. For example, to change the partition type of partition 2 to `Linux LVM`:

```
Command (m for help): t
Partition number (1-4): 2
Hex code (type L to list codes): 8e

Command (m for help): p
...
   Device Boot      Start         End      Blocks   Id  System
/dev/sdc1            2048    10487807     5242880   83  Linux
/dev/sdc2        10487808    25165823     7339008   8e  Linux LVM
```

After creating the new partition table, use the `w` command to write the table to the disk and exit `fdisk`.

```
Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.
```

If you type `q` instead, `fdisk` exits without committing the changes to disk.

For more information, see the `cfdisk(8)` and `fdisk(8)` manual pages.

Managing Partition Tables With the parted Program

> **Caution:**
>
> If any partition on the disk to be configured by using the `parted` program is currently mounted, unmount it before running `parted` on the disk. Similarly, if any partition is being used as swap space, use the `swapoff` command to disable the partition.
>
> The `parted` program is more advanced than the `fdisk` utility, as it supports more disk label types, including GPT disks, and it implements a larger set of commands.
>
> Before running the `parted` program on a disk that contains data, first back up the data onto another disk or medium.

You can use the `parted` program to do the following:

- Label a disk.
- Create a partition table.
- View an existing partition table.
- Add, modify, and delete partitions.

You can use `parted` interactively or you can specify command-line arguments. When you run `parted` interactively, you specify only the name of the disk device as an argument, for example:

```
sudo parted /dev/sda

GNU Parted 2.1
Using /dev/sda
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted)
```

The `print` command displays the partition table:

```
(parted) print

Model: ATA VBOX HARDDISK (scsi)
Disk /dev/sda: 42.9GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos

Number  Start   End     Size    Type     File system  Flags
 1      1049kB  525MB   524MB   primary  ext4         boot
 2      525MB   42.9GB  42.4GB  primary               lvm
```

The `mklabel` command creates a new partition table:

```
parted /dev/sdd
```

```
GNU Parted 2.1
Using /dev/sda
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted) mklabel
New disk label type? gpt
Warning: The existing disk label on /dev/sdd will be destroyed
and all data on this disk will be lost. Do you want to continue?
Yes/No? y

(parted) mklabel

New disk label type? gpt
Warning: The existing disk label on /dev/sdd will be destroyed
and all data on this disk will be lost. Do you want to continue?
Yes/No? y
```

Typically, you would set the disk label type to `gpt` or `msdos` for an Oracle Linux system, depending on whether the disk device supports GPT. You are prompted to confirm that you want to overwrite the existing disk label.

The `mkpart` command creates a new partition:

```
(parted) mkpart

Partition name? []? <Enter>
File system type? [ext2]? ext4
Start? 1
End? 5GB
```

For disks with an `msdos` label, you are also prompted to enter the partition type, which can be `primary`, `extended`, or `logical`. The file system type is typically set to one of the following: `fat16`, `fat32`, `ext4`, or `linux-swap` for an Oracle Linux system.

If you are going to create an `btrfs`, `ext*`, `ocfs2`, or `xfs` file system on the partition, specify `ext4` as the file system.

Unless you specify units such as GB for gigabytes, the start and end offsets of a partition are assumed to be in megabytes.

To specify the end of the disk for `End`, enter a value of `-0`.

To display the new partition, use the `print` command:

```
(parted) print

Number   Start   End     Size    File system  Name  Flags
1        1049kB  5000MB  4999MB  ext4
```

To exit the parted program, type `quit`.

> **Note:**
>
> Various `parted` commands, such as `mklabel` and `mkpart`, commit the changes to disk immediately. Unlike the `fdisk` utility, you do not have the option of quitting without saving your changes.

For more information, see the `parted(8)` manual page or type the `info parted` command to view the online user manual.

Mapping Partition Tables to Devices

You can use the `kpartx` utility to map the partitions of any block device or file that contains a partition table and partition images. `kpartx` reads the partition table and creates device files for the partitions in `/dev/mapper`. Each device file represents a disk volume or a disk partition on a device or within an image file.

The `-l` option lists any partitions that it finds, for example, in an installation image file:

```
sudo kpartx -l system.img

loop0p1 : 0 204800 /dev/loop0 2048
loop0p2 : 0 12288000 /dev/loop0 206848
loop0p3 : 0 4096000 /dev/loop0 212494848
loop0p4 : 0 2 /dev/loop0 16590848
```

This output shows that the drive image contains four partitions, and the first column are the names of the device files that can be created in `/dev/mapper`.

The `-a` option creates the device mappings:

```
sudo kpartx -a system.img
sudo ls /dev/mapper

control  loop0p1  loop0p2  loop0p3  loop0p4
```

If a partition contains a file system, you can mount it and view the files that it contains, for example:

```
sudo mkdir /mnt/sysimage
sudo mount /dev/mapper/loop0p1 /mnt/sysimage
sudo ls /mnt/sysimage

config-2.6.32-220.el6.x86_64
config-2.6.32-300.3.1.el6uek.x86_64
efi
grub
initramfs-2.6.32-220.el6.x86_64.img
initramfs-2.6.32-300.3.1.el6uek.x86_64.img
...

sudo umount /mnt/sysimage
```

The `-d` option removes the device mappings:

```
sudo kpartx -d system.img
sudo ls /dev/mapper

control
```

For more information, see the `kpartx(8)` manual page.

About Swap Space

Oracle Linux uses swap space when your system does not have enough physical memory to store the text (code) and data pages that the processes are currently using. When your system needs more memory, it writes inactive pages to swap space on disk, freeing up physical memory. However, writing to swap space has a negative impact on system performance, so increasing swap space is not an effective solution to shortage of memory. Swap space is located on disk drives, which have much slower access times than physical memory. If your system often resorts to swapping, you should add more physical memory, not more swap space.

You can configure swap space on a swap file in a file system or on a separate swap partition. A dedicated swap partition is faster, but changing the size of a swap file is easier. Configure a swap partition if you know how much swap space your system requires. Otherwise, start with a swap file and create a swap partition when you know what your system requires.

Viewing Swap Space Usage

To view a system's usage of swap space, examine the contents of `/proc/swaps`:

```
cat /proc/swaps

Filename                Type        Size      Used    Priority
/dev/sda2               partition   4128760   388     -1
/swapfile               file        999992    0       -2
```

In this example, the system is using both a 4-gigabyte swap partition on `/dev/sda2` and a one-gigabyte swap file, `/swapfile`. The `Priority` column shows that the system preferentially swaps to the swap partition rather than to the swap file.

You can also view `/proc/meminfo` or use utilities such as `free`, `top`, and `vmstat` to view swap space usage, for example:

```
grep Swap /proc/meminfo

SwapCached:          248 kB
SwapTotal:       5128752 kB
SwapFree:        5128364 kB

sudo free | grep Swap

Swap:      5128752        388    5128364
```

Creating and Using a Swap File

> **Note:**
>
> Configuring a swap file on a btrfs file system is not supported.

To create and use a swap file:

1. Use the `dd` command to create a file of the required size (for example, one million one-kilobyte blocks):

   ```
   sudo dd if=/dev/zero of=/swapfile bs=1024 count=1000000
   ```

2. Initialize the file as a swap file:

   ```
   sudo mkswap /swapfile
   ```

3. Enable swapping to the swap file:

   ```
   sudo swapon /swapfile
   ```

4. Add an entry to `/etc/fstab` for the swap file so that the system uses it following the next reboot:

   ```
   /swapfile       swap       swap       defaults       0 0
   ```

Creating and Using a Swap Partition

To create and use a swap partition:

1. Use `fdisk` to create a disk partition of type `82` (`Linux swap`) or `parted` to create a disk partition of type `linux-swap` of the size that you require.
2. Initialize the partition (for example, `/dev/sda2`) as a swap partition:

   ```
   sudo mkswap /dev/sda2
   ```

3. Enable swapping to the swap partition:

   ```
   sudo swapon /swapfile
   ```

4. Add an entry to `/etc/fstab` for the swap partition so that the system uses it following the next reboot:

   ```
   /dev/sda2       swap       swap       defaults       0 0
   ```

Removing a Swap File or Swap Partition

To remove a swap file or swap partition from use:

1. Disable swapping to the swap file or swap partition, for example:

   ```
   sudo swapoff /swapfile
   ```

2. Remove the entry for the swap file or swap partition from `/etc/fstab`.
3. Optionally, remove the swap file or swap partition if you do not want to use it in future.

2
Working With Logical Volume Manager

This chapter describes Logical Volume Manager and its use to provide data redundancy and increase performance through the implementation of volumes.

About Logical Volume Manager

You can use Logical Volume Manager (LVM) to manage multiple physical volumes and configure mirroring and striping of logical volumes to provide data redundancy and increase I/O performance. In LVM, you first create volume groups from physical volumes, which are storage devices such as disk array LUNs, software or hardware RAID devices, hard drives, and disk partitions. You can then create logical volumes in a volume group. A logical volume functions as a partition that in its implementation might be spread over multiple physical disks.

You can create file systems on logical volumes and mount the logical volume devices in the same way that you would a physical device. If a file system on a logical volume becomes full with data, you can increase the capacity of the volume by using free space in the volume group so that you can then grow the file system (provided that the file system has that capability). If necessary, you can add physical storage devices to a volume group to increase its capacity.

LVM is non-disruptive and transparent to users. You can increase the size of logical volumes and change their layout dynamically without needing to schedule system down time to reconfigure physical storage.

LVM uses the device mapper (DM) that provides an abstraction layer that enables the creation of logical devices above physical devices and provides the foundation for software RAID, encryption, and other storage features.

Initializing and Managing Physical Volumes

Before you can create a volume group, you must initialize the physical devices that you want to use as physical volumes with LVM.

> **Caution:**
>
> If the devices contain any existing data, back up the data.

To set up a physical device as a physical volume, use the `pvcreate` command:

```
sudo pvcreate [options] device ...
```

For example, set up `/dev/sdb`, `/dev/sdc`, `/dev/sdd`, and `/dev/sde` as physical volumes:

```
sudo pvcreate -v /dev/sd[bcde]
```

```
Set up physical volume for "/dev/sdb" with 6313482 available
sectors
Zeroing start of device /dev/sdb
Physical volume "/dev/sdb" successfully created
...
```

To display information about physical volumes, you can use the `pvdisplay`, `pvs`, and `pvscan` commands.

To remove a physical volume from the control of LVM, use the `pvremove` command:

```
sudo pvremove device
```

Other commands that are available for managing physical volumes include `pvchange`, `pvck`, `pvmove`, and `pvresize`.

For more information, see the `lvm(8)`, `pvcreate(8)`, and other LVM manual pages.

Creating and Managing Volume Groups

Having initialized the physical volumes, you can add them to a new or existing volume group.

To create a volume group, use the `vgcreate` command:

```
sudo vgcreate [options] volume_group physical_volume ...
```

For example, create the volume group `myvg` from the physical volumes `/dev/sdb`, `/dev/sdc`, `/dev/sdd`, and `/dev/sde`:

```
sudo vgcreate -v myvg /dev/sd[bcde]

Wiping cache of LVM-capable devices
Adding physical volume '/dev/sdb' to volume group 'myvg'
Adding physical volume '/dev/sdc' to volume group 'myvg'
Adding physical volume '/dev/sdd' to volume group 'myvg'
Adding physical volume '/dev/sde' to volume group 'myvg'
Archiving volume group "myvg" metadata (seqno 0).
Creating volume group backup "/etc/lvm/backup/myvg" (seqno 1).
Volume group "myvg" successfully created
```

LVM divides the storage space within a volume group into physical *extents*, which are the smallest unit that LVM uses when allocating storage to logical volumes. The default size of an extent is 4 MB.

The *allocation policy* for the volume group and logical volume determines how LVM allocates extents from a volume group. The default allocation policy for a volume group is `normal`, which applies rules such as not placing parallel stripes on the same physical volume. The default allocation policy for a logical volume is `inherit`, which means that the logical volume uses the same policy as for the volume group. You can change the default allocation policies by using the `lvchange` or `vgchange` commands, or you can override the allocation policy when you create a volume group or logical volume. Other allocation policies include `anywhere`, `contiguous` and `cling`.

To add physical volumes to a volume group, use the `vgextend` command:

```
sudo vgextend [options] volume_group physical_volume ...
```

To remove physical volumes from a volume group, use the `vgreduce` command:

```
sudo vgreduce [options] volume_group physical_volume ...
```

To display information about volume groups, you can use the `vgdisplay`, `vgs`, and `vgscan` commands.

To remove a volume group from LVM, use the `vgremove` command:

```
sudo vgremove volume_group
```

Other commands that are available for managing volume groups include `vgchange`, `vgck`, `vgexport`, `vgimport`, `vgmerge`, `vgrename`, and `vgsplit`.

For more information, see the `lvm(8)`, `vgcreate(8)`, and other LVM manual pages.

Creating and Managing Logical Volumes

Having create a volume group of physical volumes, you can create logical volumes from the storage space that is available in the volume group.

To create a logical volume, use the `lvcreate` command:

```
sudo lvcreate [options] --size size --name logical_volume volume_group
```

For example, create the logical volume `mylv` of size 2 GB in the volume group `myvg`:

```
sudo lvcreate -v --size 2g --name mylv myvg

Setting logging type to disk
Finding volume group "myvg"
Archiving volume group "myvg" metadata (seqno 1).
Creating logical volume mylv
Create volume group backup "/etc/lvm/backup/myvg" (seqno 2).
...
```

`lvcreate` uses the device mapper to create a block device file entry under `/dev` for each logical volume and uses udev to set up symbolic links to this device file from `/dev/mapper` and `/dev/` *volume_group*. For example, the device that corresponds to the logical volume `mylv` in the volume group `myvg` might be `/dev/dm-3`, which is symbolically linked by `/dev/mapper/myvolg-myvol` and `/dev/myvolg/myvol`.

> **Note:**
>
> Always use the devices in `/dev/mapper` or `/dev/` *volume_group*. These names are persistent and are created automatically by the device mapper early in the boot process. The `/dev/dm-*` devices are not guaranteed to be persistent across reboots.

Having created a logical volume, you can configure and use it in the same way that you would a physical storage device. For example, you can configure a logical volume as a file system, swap partition, Automatic Storage Management (ASM) disk, or raw device.

To display information about logical volumes, you can use the `lvdisplay`, `lvs`, and `lvscan` commands.

To remove a logical volume from a volume group, use the `lvremove` command:

```
sudo lvremove volume_group/logical_volume
```

> **Note:**
>
> You must specify both the name of the volume group and the logical volume.

Other commands that are available for managing logical volumes include `lvchange`, `lvconvert`, `lvmdiskscan`, `lvmsadc`, `lvmsar`, `lvrename`, and `lvresize`.

For more information, see the `lvm(8)`, `lvcreate(8)`, and other LVM manual pages.

Creating Logical Volume Snapshots

You can also use `lvcreate` with the `--snapshot` option to create a snapshot of an existing logical volume such as `mylv` in the volume group `myvg`, for example:

```
sudo lvcreate --size 500m --snapshot --name mylv-snapshot myvg/mylv

Logical volume "mylv-snapshot" created
```

You can mount and modify the contents of the snapshot independently of the original volume or preserve it as a record of the state of the original volume at the time that you took the snapshot. The snapshot usually takes up less space than the original volume, depending on how much the contents of the volumes diverge over time. In the example, we assume that the snapshot only requires one quarter of the space of the original volume. You can use the value shown by the `Snap%` column in the output from the `lvs` command to see how much data is allocated to the snapshot. If the value of `Snap%` approaches 100%, indicating that a snapshot is running out of storage, use `lvresize` to grow it. Alternatively, you can reduce a snapshot's size to save storage space. To merge a snapshot with its original volume, use the `lvconvert` command, specifying the `--merge` option.

To remove a logical volume snapshot from a volume group, use the `lvremove` command the same way that you would use the command for a logical volume:

```
sudo lvremove volume_group/logical_volume_snapshot
```

For more information, see the `lvcreate(8)` and `lvremove (8)` manual pages.

Creating and Managing Thinly-Provisioned Logical Volumes

Thinly-provisioned logical volumes have virtual sizes that are typically greater than the physical storage on which you create them. You create thinly-provisioned logical volumes from storage that you have assigned to a special type of logical volume termed a *thin pool*. LVM assigns storage on demand from a thin pool to a thinly-provisioned logical volume as required by the applications that access the volume. You need to use the `lvs` command to monitor the usage of the thin pool so that you can increase its size if its available storage is in danger of being exhausted.

To create a thin pool, use the `lvcreate` command with the `--thin` option:

```
sudo lvcreate --size size --thin volume_group/thin_pool_name
```

For example, create the thin pool `mytp` of size 1 GB in the volume group `myvg`:

```
sudo lvcreate --size 1g --thin myvg/mytp

Logical volume "mytp" created
```

You can then use `lvcreate` with the `--thin` option to create a thinly-provisioned logical volume with a size specified by the `--virtualsize` option, for example:

```
sudo lvcreate --virtualsize size --thin volume_group/thin_pool_name --name
logical_volume
```

For example, create the thinly-provisioned logical volume `mytv` with a virtual size of 2 GB using the thin pool `mytp`, whose size is currently less than the size of the volume:

```
sudo lvcreate --virtualsize 2g --thin myvg/mytp --name mytv

Logical volume "mytv" created
```

If you create a thin snapshot of a thinly-provisioned logical volume, do not specify the size of the snapshot, for example:

```
# lvcreate --snapshot --name mytv-snapshot myvg/mytv
Logical volume “mytv-snapshot” created
```

If you were to specify a size for the thin snapshot, its storage would not be provisioned from the thin pool.

If there is sufficient space in the volume group, you can use the `lvresize` command to increase the size of a thin pool, for example:

```
sudo lvresize -L+1G myvg/mytp

Extending logical volume mytp to 2 GiB
Logical volume mytp successfully resized
```

For details of how to use the `snapper` command to create and manage thin snapshots, see Using snapper with Thinly-Provisioned Logical Volumes.

For more information, see the `lvcreate(8)` and `lvresize(8)` manual pages.

Using snapper with Thinly-Provisioned Logical Volumes

You can use the `snapper` utility to create and manage thin snapshots of thinly-provisioned logical volumes.

To set up the snapper configuration for an existing mounted volume:

```
sudo snapper -c config_name create-config -f "lvm(fs_type)" fs_name
```

Here *config_name* is the name of the configuration, *fs_type* is the file system type (`ext4` or `xfs`), and *fs_name* is the path of the file system. The command adds an entry for *config_name* to `/etc/sysconfig/snapper`, creates the configuration file `/etc/snapper/configs/config_name` , and sets up a `.snapshots` subdirectory for the snapshots.

By default, `snapper` sets up a `cron.hourly` job to create snapshots in the `.snapshot` subdirectory of the volume and a `cron.daily` job to clean up old snapshots. You can edit the configuration file to disable or change this behavior. For more information, see the `snapper-configs(5)` manual page.

You can create three types of snapshots with `snapper`:

`post`
You use a *post snapshot* to record the state of a volume after a modification. A post snapshot should always be paired with a *pre snapshot* that you take immediately before you make the modification.

`pre`
You use a *pre snapshot* to record the state of a volume before a modification. A pre snapshot should always be paired with a *post snapshot* that you take immediately after you have completed the modification.

`single`
You can use a single snapshot to record the state of a volume but it does not have any association with other snapshots of the volume.

For example, the following commands create pre and post snapshots of a volume:

```
sudo snapper -c config_name create -t pre -p N

... Modify the volume's contents ...

sudo snapper -c config_name create -t post --pre-num N p N'
```

The `-p` option causes `snapper` to display the number of the snapshot so that you can reference it when you create the post snapshot or when you compare the contents of the pre and post snapshots.

To display the files and directories that have been added, removed, or modified between the pre and post snapshots, use the `status` subcommand:

```
sudo snapper -c config_name status N .. N'
```

To display the differences between the contents of the files in the pre and post snapshots, use the `diff` subcommand:

```
sudo snapper -c config_name diff N .. N'
```

To list the snapshots that exist for a volume:

```
sudo snapper -c config_name list
```

To delete a snapshot, specify its number to the `delete` subcommand:

```
sudo snapper -c config_name delete N''
```

To undo the changes in the volume from post snapshot *N'* to pre snapshot *N*:

```
sudo snapper -c config_name undochange N..N'
```

For more information, see the `snapper(8)` manual page.

3
Working With Software RAID

This chapter describes RAID features, with special focus on the use of software RAID for storage redundancy.

About Software RAID

The Redundant Array of Independent Disks (RAID) feature enables you to spread data across the drives to increase capacity, implement data redundancy, and increase performance. RAID is usually implemented either in hardware on intelligent disk storage that exports the RAID volumes as LUNs, or in software by the operating system. Oracle Linux kernel uses the multidisk (MD) driver to support software RAID by creating virtual devices from two or more physical storage devices. You can use MD to organize disk drives into RAID devices and implement different RAID levels.

The following software RAID levels are commonly used with Oracle Linux:

Linear RAID (spanning)
Combines drives as a larger virtual drive. There is no data redundancy or performance benefit. Resilience decreases because the failure of a single drive renders the array unusable.

RAID-0 (striping)
Increases performance but does not provide data redundancy. Data is broken down into units (stripes) and written to all the drives in the array. Resilience decreases because the failure of a single drive renders the array unusable.

RAID-1 (mirroring)
Provides data redundancy and resilience by writing identical data to each drive in the array. If one drive fails, a mirror can satisfy I/O requests. Mirroring is an expensive solution because the same information is written to all of the disks in the array.

RAID-5 (striping with distributed parity)
Increases read performance by using striping and provides data redundancy. The parity is distributed across all the drives in an array but it does not take up as much space as a complete mirror. Write performance is reduced to some extent from RAID-0 by having to calculate parity information and write this information in addition to the data. If one disk in the array fails, the parity information is used to reconstruct data to satisfy I/O requests. In this mode, read performance and resilience are degraded until you replace the failed drive and it is repopulated with data and parity information. RAID-5 is intermediate in expense between RAID-0 and RAID-1.

RAID-6 (striping with double distributed parity)
A more resilient variant of RAID-5 that can recover from the loss of two drives in an array. RAID-6 is used when data redundancy and resilience are important, but performance is not. RAID-6 is intermediate in expense between RAID-5 and RAID-1.

RAID 0+1 (mirroring of striped disks)
Combines RAID-0 and RAID-1 by mirroring a striped array to provide both increased performance and data redundancy. Failure of a single disk causes one of the mirrors to be unusable until you replace the disk and repopulate it with data. Resilience is degraded while only a single mirror remains available. RAID 0+1 is usually as expensive as or slightly more expensive than RAID-1.

RAID 1+0 (striping of mirrored disks or RAID-10)
Combines RAID-0 and RAID-1 by striping a mirrored array to provide both increased performance and data redundancy. Failure of a single disk causes part of one mirror to be unusable until you replace the disk and repopulate it with data. Resilience is degraded while only a single mirror retains a complete copy of the data. RAID 1+0 is usually as expensive as or slightly more expensive than RAID-1.

Creating Software RAID Devices

To create a software RAID device:

1. Use the `mdadm` command to create the MD RAID device:

   ```
   sudo mdadm --create md_device --level=RAID_level [options] --raid-devices=N
   device ...
   ```

   For example, to create a RAID-1 device `/dev/md0` from `/dev/sdf` and `/dev/sdg`:

   ```
   sudo mdadm --create /dev/md0 --level=1 -raid-devices=2 /dev/sd[fg]
   ```

   Create a RAID-5 device `/dev/md1` from `/dev/sdb`, `/dev/sdc`, and `dev/sdd`:

   ```
   sudo mdadm --create /dev/md1 --level=5 -raid-devices=3 /dev/sd[bcd]
   ```

   If you want to include spare devices that are available for expansion, reconfiguration, or replacing failed drives, use the `--spare-devices` option to specify their number, for example:

   ```
   sudo mdadm --create /dev/md1 --level=5 -raid-devices=3 --spare-
   devices=1 /dev/sd[bcde]
   ```

   > **Note:**
   >
   > The number of RAID and spare devices must equal the number of devices that you specify.

2. Add the RAID configuration to `/etc/mdadm.conf`:

   ```
   # mdadm --examine --scan >> /etc/mdadm.conf
   ```

   > **Note:**
   >
   > This step is optional. It helps `mdadm` to assemble the arrays at boot time.

   For example, the following entries in `/etc/mdadm.conf` define the devices and arrays that correspond to `/dev/md0` and `/dev/md1`:

```
DEVICE /dev/sd[c-g]
ARRAY /dev/md0 devices=/dev/sdf,/dev/sdg
ARRAY /dev/md1 spares=1 devices=/dev/sdb,/dev/sdc,/dev/sdd,/dev/sde
```

For more examples, see the sample configuration file `/usr/share/doc/mdadm-3.2.1/mdadm.conf-example`.

Having created an MD RAID device, you can configure and use it in the same way that you would a physical storage device. For example, you can configure it as an LVM physical volume, file system, swap partition, Automatic Storage Management (ASM) disk, or raw device.

You can view `/proc/mdstat` to check the status of the MD RAID devices, for example:

```
cat /proc/mdstat

Personalities : [raid1]
mdo : active raid1 sdg[1] sdf[0]
```

To display summary and detailed information about MD RAID devices, you can use the `--query` and `--detail` options with `mdadm`.

For more information, see the `md(4)`, `mdadm(8)`, and `mdadm.conf(5)` manual pages.

4
Creating Encrypted Block Devices

This chapter describes how to use encrypted block devices to secure stored data.

About Encrypted Block Devices

The device mapper supports the creation of encrypted block devices using the `dm-crypt` device driver. You can access data on encrypted devices at boot time only if you enter the correct password. As the underlying block device is encrypted and not the file system, you can use `dm-crypt` to encrypt disk partitions, RAID volumes, and LVM physical volumes, regardless of their contents.

When you install Oracle Linux, you have the option of configure encryption on system volumes other than the partition from which the system boots. If you want to protect the bootable partition, consider using any password protection mechanism that is built into the BIOS or setting up a GRUB password.

Setting Up LUKS Encryption

You use the `cryptsetup` utility to set up Linux Unified Key Setup (LUKS) encryption on the device and to manage authentication.

To set up the mapped device for an encrypted volume:

1. Initialize a LUKS partition on the device and set up the initial key, for example:

```
sudo cryptsetup luksFormat /dev/sdd

WARNING!
========
This will overwrite data on /dev/sdd irrevocably.
Are you sure? (Type uppercase yes): YES
Enter LUKS passphrase: passphrase
Verify passphrase: passphrase
```

2. Open the device and create the device mapping:

```
sudo cryptsetup luksOpen /dev/sdd cryptfs

Enter passphrase for /dev/sdd: passphrase
```

   In this example, the encrypted volume is accessible as `/dev/mapper/cryptfs`.

3. Create an entry for the encrypted volume in `/etc/crypttab`, for example:

```
# <target name>  <source device>  <key file>  <options>
cryptfs          /dev/sdd         none        luks
```

   This entry causes the operating system to prompt you to enter the passphrase at boot time.

After you have created an encrypted volume and its device mapping, you can then configure and use it in the same way that you would a physical storage device. For example, you can

configure it as an LVM physical volume, file system, swap partition, Automatic Storage Management (ASM) disk, or raw device. For example, you would create an entry in the `/etc/fstab` to mount the mapped device (`/dev/mapper/cryptfs`), not the physical device (`/dev/sdd`).

To verify the status of an encrypted volume, use the following command:

```
sudo cryptsetup status cryptfs

/dev/mapper/cryptfs is active.
type: LUKS1
cipher:  aes-cbs-essiv:sha256
keysize: 256 bits
device:  /dev/xvdd1
offset:  4096 sectors
size:    6309386 sectors
mode:    read/write
```

Should you need to remove the device mapping, unmount any file system that the encrypted volume contains, and run the following command:

```
sudo cryptsetup luksClose /dev/mapper/cryptfs
```

For more information, see the `cryptsetup(8)` and `crypttab(5)` man pages.

Recommendations for SSD Configuration for Btrfs, `ext4`, and Swap

When partitioning an SSD, align primary and logical partitions on one-megabyte (1048576 bytes) boundaries. If partitions, file system blocks, or RAID stripes are incorrectly aligned and overlap the boundaries of the underlying storage's pages, which are usually either 4 KB or 8 KB in size, the device controller has to modify twice as many pages than if correct alignment is used.

For btrfs and ext4 file systems, specifying the `discard` option with `mount` sends discard (TRIM) commands to an underlying SSD whenever blocks are freed. This option can extend the working life of the device but it has a negative impact on performance, even for SSDs that support queued discards. The recommended alternative is to use the `fstrim` command to discard empty blocks that the file system is not using, especially before reinstalling the operating system or before creating a new file system on an SSD. Schedule `fstrim` to run when it will have minimal impact on system performance. You can also apply `fstrim` to a specific range of blocks rather than the whole file system.

> **Note:**
>
> Using a minimal journal size of 1024 file-system blocks for ext4 on an SSD improves performance. However, it is not recommended that you disable journalling altogether as it improves the robustness of the file system.

Btrfs automatically enables SSD optimization for a device if the value of `/sys/block/device/queue/rotational` is 0. If btrfs does not detect a device as being an SSD, you can enable SSD optimization by specifying the `ssd` option to `mount`.

> **Note:**
>
> By default, btrfs enables SSD optimization for Xen Virtual Devices (XVD) because the value of `rotational` for these devices is 0. To disable SSD optimization, specify the `nossd` option to `mount`.
>
> Setting the `ssd` option does not imply that `discard` is also set.

If you configure swap files or partitions on an SSD, reduce the tendency of the kernel to perform anticipatory writes to swap, which is controlled by the value of the `vm.swappiness` kernel parameter and displayed as `/proc/sys/vm/swappiness`. The value of `vm.swappiness` can be in the range 0 to 100, where a higher value implies a greater propensity to write to swap. The default value is 60. The suggested value when swap has been configured on SSD is 1. You can use the following commands to change the value:

```
sudo echo "vm.swappiness = 1" >> /etc/sysctl.conf
sudo sysctl -p

...
vm.swappiness = 1
```

5
Working With iSCSI Devices

This chapter discusses using iSCSI devices for data storage.

About Linux-IO Storage Configuration

Oracle Linux 7 with both UEK and RHCK uses the Linux-IO Target (LIO) to provide the block-storage SCSI target for FCoE, iSCSI, and Mellanox InfiniBand (iSER and SRP). You can manage LIO by using the `targetcli` shell provided in the `targetcli` package. Note that Mellanox InfiniBand is only supported with UEK.

Fibre Channel over Ethernet (FCoE) encapsulates Fibre Channel packets in Ethernet frames, which allows the packets to be sent over Ethernet networks. To configure FCoE storage, you also need to install the `fcoe-utils` package, which provides the `fcoemon` service and the `fcoeadm` command.

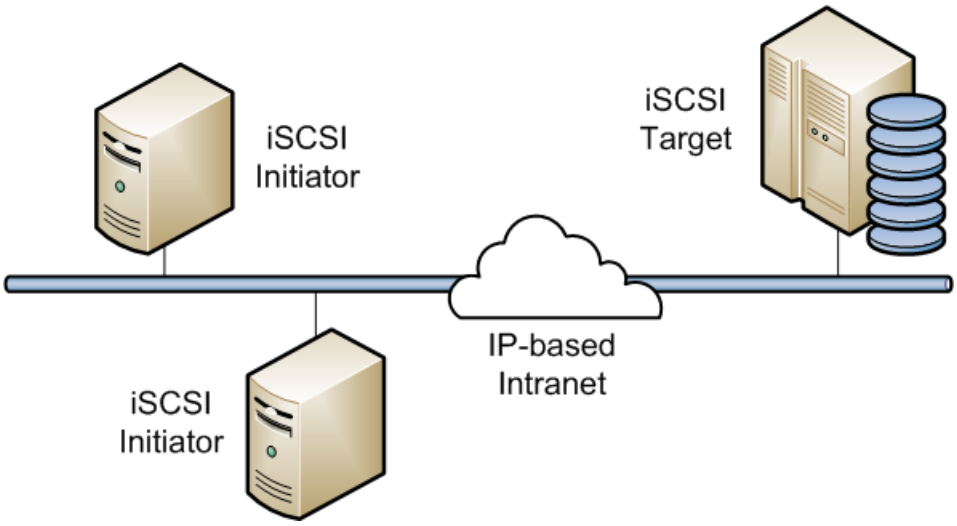
The Internet Small Computer System Interface (iSCSI) is an IP-based standard for connecting storage devices. iSCSI encapsulates SCSI commands in IP network packets, which allows data transfer over long distances and sharing of storage by client systems. As iSCSI uses the existing IP infrastructure, it does not require the purchase and installation of fiber-optic cabling and interface adapters that are needed to implement Fibre Channel (FC) storage area networks.

A client system (*iSCSI initiator*) accesses the storage server (*iSCSI target*) over an IP network. To an iSCSI initiator, the storage appears to be locally attached.

An iSCSI target is typically a dedicated, network-connected storage device but it can also be a general-purpose computer.

Figure 5-1 shows a simple network where several iSCSI initiators are able to access the shared storage that is attached to an iSCSI target.

Figure 5-1 iSCSI Initiators and an iSCSI Target Connected via an IP-based Network



A hardware-based iSCSI initiator uses a dedicated iSCSI HBA. Oracle Linux supports iSCSI initiator functionality in software. The kernel-resident device driver uses the existing network interface card (NIC) and network stack to emulate a hardware iSCSI initiator. As the iSCSI initiator functionality is not available at the level of the system BIOS, you cannot boot an Oracle Linux system from iSCSI storage .

To improve performance, some network cards implement TCP/IP Offload Engines (TOE) that can create a TCP frame for the iSCSI packet in hardware. Oracle Linux does not support TOE, although suitable drivers may be available directly from some card vendors.

For more information about LIO, see http://linux-iscsi.org/wiki/Main_Page.

Configuring an iSCSI Target

The following procedure describes how to set up a basic iSCSI target on an Oracle Linux system by using block storage backends. Note that you can use other storage backend types to set up an iSCSI target.

When you use the `targetcli` command, it saves the current configuration to the JSON-format file `/etc/target/saveconfig.json`. See the `targetcli(8)` manual page for additional information.

1. Run the `targetcli` interactive shell:

```
sudo targetcli

targetcli shell version 2.1.fb31
Copyright 2011-2013 by Datera, Inc and others.
For help on commands, type 'help'.

/>
```

   At the interactive shell prompt, list the object hierarchy, which is initially empty:

```
/> ls

o- / ..........................................................................
[...]
  o- backstores ...............................................................
[...]
  | o- block ..................................... [Storage
Objects: 0]
  | o- fileio .................................... [Storage
Objects: 0]
  | o- pscsi ..................................... [Storage
Objects: 0]
  | o- ramdisk ................................... [Storage
Objects: 0]
  o- iscsi .....................................................
[Targets: 0]
  o- loopback ..................................................
[Targets: 0]
```

2. Change to the `/backstores/block` directory and create a block storage object for the disk partitions that you want to provide as LUNs, for example:

```
/> cd /backstores/block
/backstores/block> create name=LUN_0 dev=/dev/sdb
Created block storage object LUN_0 using /dev/sdb.
```

```
/backstores/block> create name=LUN_1 dev=/dev/sdc
Created block storage object LUN_1 using /dev/sdc.
```

The names that you assign to the storage objects are arbitrary.

> **Note:**
>
> The device path varies, based on the Oracle Linux instance's disk configuration.

3. Change to the `/iscsi` directory and create an iSCSI target:

```
/> cd /iscsi
/iscsi> create

Created target iqn.2013-01.com.mydom.host01.x8664:sn.ef8e14f87344.
Created TPG 1.
```

List the target portal group (TPG) hierarchy, which is initially empty:

```
/iscsi> ls

o- iscsi ..................................................... [Targets: 1]
  o- iqn.2013-01.com.mydom.host01.x8664:sn.ef8e14f87344 ..... [TPGs: 1]
    o- tpg1 ....................................... [no-gen-acls, no-auth]
      o- acls ..................................................... [ACLs: 0]
      o- luns ..................................................... [LUNs: 0]
      o- portals ................................................. [Portals: 0]
```

4. Change to the `luns` subdirectory of the TPG directory hierarchy and add the LUNs to the target portal group:

```
/iscsi> cd iqn.2013-01.com.mydom.host01.x8664:sn.ef8e14f87344/tpg1/luns
/iscsi/iqn.20...344/tpg1/luns> create /backstores/block/LUN_0
Created LUN 0.
/iscsi/iqn.20...344/tpg1/luns> create /backstores/block/LUN_1
Created LUN 1.
```

5. Change to the `portals` subdirectory of the TPG directory hierarchy and specify the IP address and TCP port of the iSCSI endpoint:

```
/iscsi/iqn.20...344/tpg1/luns> cd ../portals
/iscsi/iqn.20.../tpg1/portals> create 10.150.30.72 3260
Using default IP port 3260
Created network portal 10.150.30.72:3260.
```

If you omit the TCP port number, the default value is `3260`.

> **Note:**
>
> If the portal creation fails, and a message similar to the following is displayed, it is most likely because a default portal was previously created:
>
> ```
> Could not create NetworkPortal in configFS
> ```
>
> In this case, first delete the default portal, then repeat Step 5 to create the new portal.
>
> ```
> /iscsi/iqn.20.../tpg1/portals> delete 0.0.0.0 ip_port=3260
> ```

6. Enable TCP port 3260 by running either of the following commands:

```
sudo firewall-cmd --permanent --add-port=3260/tcp

sudo firewall-cmd --permanent --add-service iscsi-target
```

7. List the object hierarchy, which now shows the configured block storage objects and TPG:

```
/iscsi/iqn.20.../tpg1/portals> ls /
o- / ..........................................................................
[...]
  o- backstores ...............................................................
[...]
  | o- block ..................................... [Storage
Objects: 1]
  | | o- LUN_0 ....................... [/dev/sdb (10.0GiB) write-thru
activated]
  | | o- LUN_1 ....................... [/dev/sdc (10.0GiB) write-thru
activated]
  | o- fileio ..................................... [Storage
Objects: 0]
  | o- pscsi ...................................... [Storage
Objects: 0]
  | o- ramdisk .................................... [Storage
Objects: 0]
  o- iscsi ...................................................
[Targets: 1]
  | o- iqn.2013-01.com.mydom.host01.x8664:sn.ef8e14f87344 ............
[TPGs: 1]
  |   o- tpg1 ....................................... [no-gen-acls, no-
auth]
  |     o- acls ................................................
[ACLs: 0]
  |     o- luns ................................................
[LUNs: 1]
  |     | o- lun0 ................................... [block/LUN_0 (/dev/
sdb)]
  |     | o- lun1 ................................... [block/LUN_1 (/dev/
sdc)]
  |     o- portals .............................................
[Portals: 1]
  |       o- 10.150.30.72:3260 ........................................
[OK]
  o- loopback ................................................
[Targets: 0]
```

8. Configure the access rights for logins by initiators. For example, to configure demonstration mode that does not require authentication, change to the TGP directory and set the values of the `authentication` and `demo_mode_write_protect` attributes to 0 and `generate_node_acls cache_dynamic_acls` to 1:

```
/iscsi/iqn.20.../tpg1/portals> cd ..
/iscsi/iqn.20...14f87344/tpg1> set attribute authentication=0
demo_mode_write_protect=0
                       generate_node_acls=1 cache_dynamic_acls=1
Parameter authentication is now '0'.
Parameter demo_mode_write_protect is now '0'.
Parameter generate_node_acls is now '1'.
Parameter cache_dynamic_acls is now '1'.
```

> **Caution:**
>
> Demonstration mode is inherently insecure. For information about configuring secure authentication modes, see http://linux-iscsi.org/wiki/ISCSI#Define_access_rights.

9. Change to the `root` directory and save the configuration.

```
/iscsi/iqn.20...14f87344/tpg1> cd /
/> saveconfig
Last 10 configs saved in /etc/target/backup.
Configuration saved to /etc/target/saveconfig.json
```

10. Enable the target service.

```
# systemctl enable target.service
```

> **Note:**
>
> Saving the configuration and enabling the target service ensures that the changes persist across system reboots. Note also that failure to run these commands can result in an empty configuration.

Restoring a Saved Configuration for an iSCSI target

To restore a saved configuration for an iSCSI target, start the `targetcli` interactive shell and then run the following command:

```
sudo targetcli

targetcli shell version 2.1.fb46
Copyright 2011-2013 by Datera, Inc and others.
For help on commands, type 'help'.

/>
```

At the interactive shell prompt, type the appropriate command, for example:

```
/> restoreconfig /etc/target/saveconfig.json
```

In the previous example, `/etc/target/saveconfig.json` is the most recently saved configuration.

Or, you can run the following command to restore saved configurations from previous versions:

```
/> restoreconfig /etc/target/backup/saveconfig-20180516-18:53:29.json
```

Configuring an iSCSI Initiator

To configure an Oracle Linux system as an iSCSI initiator:

1. Install the `iscsi-initiator-utils` package:

   ```
   sudo yum install iscsi-initiator-utils
   ```

2. Use a discovery method to discover the iSCSI targets at the specified IP address.

   You can use either the `SendTargets` or the Internet Storage Name Service (iSNS) discovery method.

   For example, you would use the `SendTargets` discovery method as follows:

   ```
   # iscsiadm -m discovery -t sendtargets -p 10.150.30.72

   10.150.30.72:3260,1 iqn.2013-01.com.mydom.host01.x8664:sn.ef8e14f87344
   ```

   This command also starts the `iscsid` service if it is not already running.

   - If the network connection fails, adjust the firewall settings for your network to allow communication with the iSCSI target:

     ```
     sudo iptables -D INPUT -j REJECT --reject-with icmp-host-prohibited
     sudo iptables -D FORWARD -j REJECT --reject-with icmp-host-prohibited
     ```

   - Then, re-run the command using the `SendTargets` discovery method:

     ```
     sudo iscsiadm -m discovery -t sendtargets -p 10.150.30.72

     10.150.30.72:3260,1 iqn.2013-01.com.mydom.host01.x8664:sn.ef8e14f87344
     ```

3. Display information about the targets that are now stored in the discovery database by running the following command:

   ```
   sudo iscsiadm -m discoverydb -t st -p 10.150.30.72

   # BEGIN RECORD 6.2.0.873-14
   discovery.startup = manual
   discovery.type = sendtargets
   discovery.sendtargets.address = 10.150.30.72
   discovery.sendtargets.port = 3260
   discovery.sendtargets.auth.authmethod = None
   discovery.sendtargets.auth.username = <empty>
   discovery.sendtargets.auth.password = <empty>
   discovery.sendtargets.auth.username_in = <empty>
   discovery.sendtargets.auth.password_in = <empty>
   discovery.sendtargets.timeo.login_timeout = 15
   discovery.sendtargets.use_discoveryd = No
   discovery.sendtargets.discoveryd_poll_inval = 30
   discovery.sendtargets.reopen_max = 5
   discovery.sendtargets.timeo.auth_timeout = 45
   discovery.sendtargets.timeo.active_timeout = 30
   ```

```
discovery.sendtargets.iscsi.MaxRecvDataSegmentLength = 32768
# END RECORD
```

4. Establish a session and log in to a specific target:

```
sudo iscsiadm -m node -T iqn.2013-01.com.mydom.host01.x8664:sn.ef8e14f87344 -p
10.150.30.72:3260 -l

Login to [iface: default, target: iqn.2003-01.org.linux-iscsi.localhost.x8664:
sn.ef8e14f87344, portal: 10.150.30.72,3260] successful.
```

5. Verify that the session is active and display the available LUNs:

```
sudo iscsiadm -m session -P 3

iSCSI Transport Class version 2.0-870
version 6.2.0.873-14
Target: iqn.2003-01.com.mydom.host01.x8664:sn.ef8e14f87344 (non-flash)
    Current Portal: 10.0.0.2:3260,1
    Persistent Portal: 10.0.0.2:3260,1
        **********
        Interface:
        **********
        Iface Name: default
        Iface Transport: tcp
        Iface Initiatorname: iqn.1994-05.com.mydom:ed7021225d52
        Iface IPaddress: 10.0.0.2
        Iface HWaddress: <empty>
        Iface Netdev: <empty>
        SID: 5
        iSCSI Connection State: LOGGED IN
        iSCSI Session State: LOGGED_IN
        Internal iscsid Session State: NO CHANGE
.
.
.
        ************************
        Attached SCSI devices:
        ************************
        Host Number: 8    State: running
        scsi8 Channel 00 Id 0 Lun: 0
            Attached scsi disk sdb        State: running
        scsi8 Channel 00 Id 0 Lun: 1
            Attached scsi disk sdc        State: running
```

The LUNs are represented as SCSI block devices (`sd*`) in the local `/dev` directory, for example:

```
sudo fdisk -l | grep /dev/sd[bc]

Disk /dev/sdb: 10.7 GB, 10737418240 bytes, 20971520 sectors
Disk /dev/sdc: 10.7 GB, 10737418240 bytes, 20971520 sectors
```

To distinguish between target LUNs, examine the paths under `/dev/disk/by-path`:

```
ls -l /dev/disk/by-path/

lrwxrwxrwx  1 root root  9 May 15 21:05
  ip-10.150.30.72:3260-iscsi-iqn.2013-01.com.mydom.host01.x8664:
  sn.ef8e14f87344-lun-0 -> ../../sdb
lrwxrwxrwx 1 root root  9 May 15 21:05
  ip-10.150.30.72:3260-iscsi-iqn.2013-01.com.mydom.host01.x8664:
  sn.ef8e14f87344-lun-1 -> ../../sdc
```

You can view the initialization messages for the LUNs in the `/var/log/messages` file as follows:

```
sudo grep sdb /var/log/messages

...
May 18 14:19:36 localhost kernel: [12079.963376] sd 8:0:0:0: [sdb] Attached
SCSI disk
...
```

You can configure and use a LUN in the same way that you would any other physical storage device, for example, as an LVM physical volume, a file system, a swap partition, an Automatic Storage Management (ASM) disk, or a raw device.

Specify the `_netdev` option when creating mount entries for iSCSI LUNs in `/etc/fstab`, for example:

```
UUID=084591f8-6b8b-c857-f002-ecf8a3b387f3     /iscsi_mount_point
ext4     _netdev    0  0
```

This option indicates the file system resides on a device that requires network access, and prevents the system from attempting to mount the file system until the network has been enabled.

> **Note:**
>
> Specify an iSCSI LUN in `/etc/fstab` by using `UUID=` *UUID* rather than the device path. A device path can change after re-connecting the storage or rebooting the system. You can use the `blkid` command to display the `UUID` of a block device.
>
> Any discovered LUNs remain available across reboots provided that the target continues to serve those LUNs and you do not log the system off the target.

For more information, see the `iscsiadm(8)` and `iscsid(8)` manual pages.

Updating the Discovery Database

If the LUNs that are available on an iSCSI target change, you can use the `iscsiadm` command on an iSCSI initiator to update the entries in its discovery database. The following example assume that the target supports the SendTargets discovery method

To add new records that are not currently in the database:

```
sudo iscsiadm --mode discoverydb -type st -p 10.150.30.72 -o new --discover
```

To update existing records in the database:

```
sudo iscsiadm -m discoverydb -t st -p 10.150.30.72 -o update --discover
```

To delete records from the database that are no longer supported by the target:

```
sudo iscsiadm -m discoverydb -t st -p 10.150.30.72 -o delete --discover
```

For more information, see the `iscsiadm(8)` manual page.

6
Using Multipathing for Efficient Storage

Multiple paths to storage devices provide connection redundancy, failover capability, load balancing, and improved performance. Device-Mapper Multipath (DM-Multipath) is a multipathing tool that enables you represent multiple I/O paths between a server and a storage device as a single path.

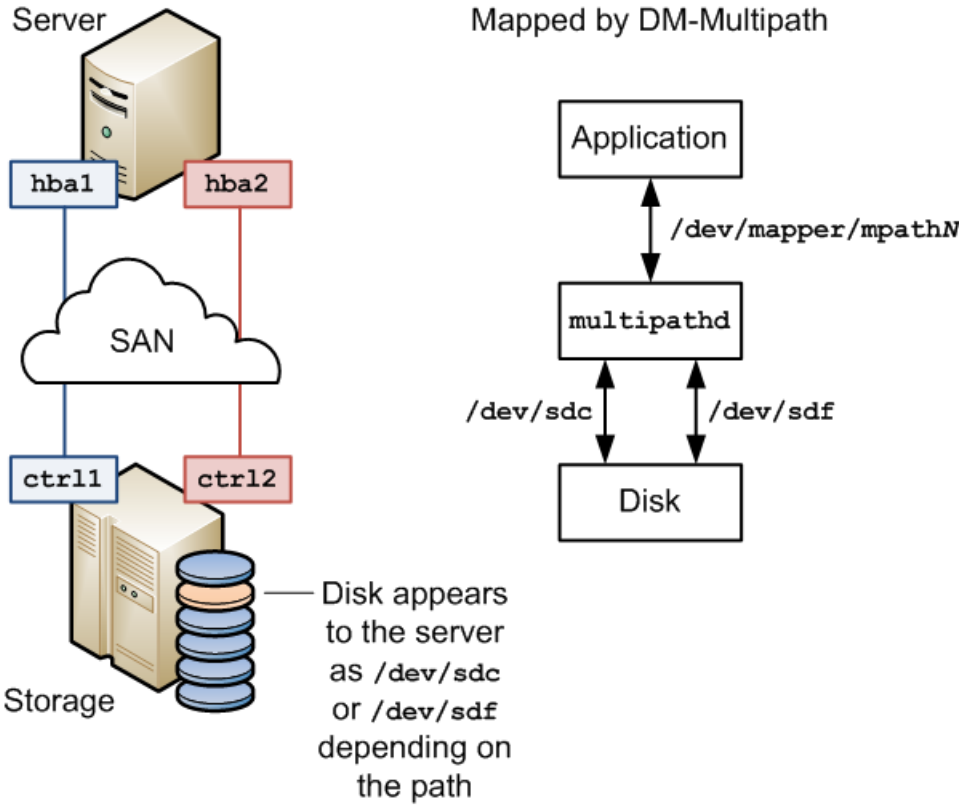
Device Multipathing Sample Setup

You would be most likely to configure multipathing with a system that can access storage on a Fibre Channel-based storage area network (SAN). You can also use multipathing on an iSCSI initiator if redundant network connections exist between the initiator and the target.

Figure 6-1 shows a simple DM-Multipath configuration where two I/O paths are configured between a server and a disk on a SAN-attached storage array:

- Between host bus adapter `hba1` on the server and controller `ctrl1` on the storage array.
- Between host bus adapter `hba2` on the server and controller `ctrl2` on the storage array.

Figure 6-1 DM-Multipath Mapping of Two Paths to a Disk over a SAN



Without DM-Multipath, the system treats each path as being separate even though it connects the server to the same storage device. DM-Multipath creates a single multipath device, `/dev/mapper/mpathN` , that subsumes the underlying devices, `/dev/sdc` and `/dev/sdf`.

You can configure the multipathing service (`multipathd`) to handle I/O from and to a multipathed device in one of the following ways:

Active/Active
I/O is distributed across all available paths, either by round-robin assignment or dynamic load-balancing.

Active/Passive (standby failover)
I/O uses only one path. If the active path fails, DM-Multipath switches I/O to a standby path. This is the default configuration.

> **Note:**
>
> DM-Multipath can provide failover in the case of path failure, such as in a SAN fabric. Disk media failure must be handled by using either a software or hardware RAID solution.

Configuring Multipathing

The following procedure describes how to set up a simple multipath configuration on a server with access to SAN-attached storage.

1. Install the `device-mapper-multipath` package.

   ```
   sudo yum install device-mapper-multipath
   ```

2. Initiate the basic configuration settings of the multipathing feature.

   ```
   sudo mpathconf --enable --with_multipathd y
   ```

   This command also creates the `/etc/multipath.conf` file.

3. (Optional) To determine the status of multipathing, use the following command:

   ```
   sudo mpathconf
   ```

4. Edit `/etc/multipath.conf`, as required.

   For details, see Working With the Multipathing Configuration File.

   If you want to display the current multipath configuration, use the `multipath -ll` command, for example:

   ```
   sudo multipath -ll
   ```

   The command displays output similar to the following, when multipath is configured properly:

   ```
   mpath1(360000970000292602744533030303730) dm-0 SUN,(StorEdge 3510|T4
   size=20G features='0' hwhandler='0' wp=rw
   |-+- policy='round-robin 0' prio=1 status=active
   | `- 5:0:0:2 sdb 8:16    active ready running
   ```

```
`-+- policy='round-robin 0' prio=1 status=active
  `- 5:0:0:3 sdc 8:32    active ready running
```

The previous output shows that `/dev/mapper/mpath1` subsumes two paths (`/dev/sdb` and `/dev/sdc`) to 20 GB of storage in an active/active configuration using the round-robin I/O path selection. The WWID that identifies the storage is `3600009700002926027445330303037 30` and the name of the multipath device under `sysfs` is `dm-0`.

For more information, see the `mpathconf(8)`, `multipath(8)`, `multipathd(8)`, `multipath.conf(5)`, and `scsi_id(8)` manual pages.

Working With the Multipathing Configuration File

By using the `/etc/multipath.conf` file, you can add a combination of definitions that customizes multipathing according to your system environment setup. You can obtain a commented example configuration from `/usr/share/doc/device-mapper-multipath/multipath.conf`.

The `/etc/multipath.conf` file is divided into the following typical sections:

`defaults`
Defines default multipath settings, which can be overridden by settings in the `devices` section, and which in turn can be overridden by settings in the `multipaths` section.

`blacklist`
Defines devices that are excluded from multipath topology discovery. Blacklisted devices cannot subsumed by a multipath device.
The example shows the three ways that you can use to exclude devices: by WWID (`wwid`), by device name (`devnode`), and by device type (`device`).

`blacklist_exceptions`
Defines devices that are included in multipath topology discovery, even if the devices are implicitly or explicitly listed in the `blacklist` section.

`multipaths`
Defines settings for a multipath device that is identified by its WWID.
The `alias` attribute specifies the name of the multipath device as it will appear in `/dev/mapper` instead of a name based on either the WWID or the multipath group number.
To obtain the WWID of a SCSI device, use the `scsi_id` command:

```
sudo scsi_id --whitelisted --replace-whitespace --device=device_name
```

`devices`
Defines settings for individual types of storage controller. Each controller type is identified by the `vendor`, `product`, and optional `revision` settings, which must match the information in `sysfs` for the device.
You can find details of the storage arrays that DM-Multipath supports and their default configuration values in `/usr/share/doc/device-mapper-multipath-version/multipath.conf.defaults`, which you can use as the basis for entries in the `/etc/multipath.conf` file.
To add a storage device that DM-Multipath does not list as being supported, obtain the vendor, product, and revision information from the `vendor`, `model`, and `rev` files under `/sys/block/device_name/device`.

The following entries in the `/etc/multipath.conf` file would be appropriate for setting up active/passive multipathing to an iSCSI LUN with the specified WWID.

```
defaults {
    user_friendly_names    yes
    uid_attribute          ID_SERIAL
}

multipaths {
    multipath {
        wwid 360000970000292602744533030303730
    }
}
```

In this standby failover configuration, I/O continues through a remaining active network interface if a network interfaces fails on the iSCSI initiator.

> **Note:**
>
> If you edit the `/etc/multipath.conf` file, restart the `multipathd` service to make so that the file is re-read.

For more information about configuring entries in `/etc/multipath.conf`, refer to the `multipath.conf(5)` manual page.