Oracle® Linux 8
Managing Storage Devices
Abstract

*Oracle® Linux 8: Using OpenSCAP for Security Compliance* describes tasks for using OpenSCAP to scan your Oracle Linux system for security vulnerabilities to make them comply to security standards.

Document generated on: 2020-03-18 (revision: 9498)
# Table of Contents

Preface ........................................................................................................................................................................................................... v  
1 Configuring Storage Drives ........................................................................................................................................................................... 1  
   1.1 Using Disk Partitions ........................................................................................................................................................................... 1  
      1.1.1 About Disk Partitions ................................................................................................................................................................. 1  
      1.1.2 Partitioning Disks by Using fdisk ............................................................................................................................................... 2  
      1.1.3 Partitioning Disks by Using parted ........................................................................................................................................... 4  
      1.1.4 Mapping Partition Tables to Devices ......................................................... 6  
   1.2 Implementing Swap Spaces ................................................................................................................................................................. 7  
      1.2.1 About Swap Space ...................................................................................................................................................................... 8  
      1.2.2 Creating a Swap File ................................................................................................................................................................. 8  
      1.2.3 Creating a Swap Partition .......................................................................................................................................................... 9  
      1.2.4 Viewing Swap Space Usage .................................................................................................................................................... 9  
      1.2.5 Removing a Swap File or Swap Partition .................................................. 9  
   1.3 Recommendations for Solid State Drives ................................................................. 9  
2 Working With Logical Volume Manager .............................................................................................................................. 11  
   2.1 About Logical Volume Manager .................................................................................................................................................. 11  
   2.2 Setting Up Logical Volumes ......................................................................................................................................................... 11  
      2.2.1 Initializing and Managing Physical Volumes .................................................. 12  
      2.2.2 Creating and Managing Volume Groups ........................................................ 12  
      2.2.3 Creating and Managing Logical Volumes ...................................................... 13  
      2.2.4 Creating Logical Volume Snapshots ............................................................ 13  
   2.3 Using Thinly-Provisioned Logical Volumes ............................................................ 14  
      2.3.1 Configuring and Managing Thinly-Provisioned Logical Volumes ................. 14  
      2.3.2 Using snapper With Thinly-Provisioned Logical Volumes ......................... 15  
3 Working With Software RAID .................................................................................................................................................... 17  
   3.1 About Software RAID ............................................................................................................................................................... 17  
   3.2 Creating Software RAID Devices ........................................................................ 18  
4 Using Encrypted Block Devices .................................................................................................................................................. 21  
   4.1 About Encrypted Block Devices ........................................................................... 21  
   4.2 Creating Encrypted Volumes .............................................................................. 21  
5 Working With iSCSI Devices ................................................................................................................................................... 23  
   5.1 About Linux-IO Storage Configuration .............................................................. 23  
   5.2 Configuring an iSCSI Target ................................................................................. 24  
   5.3 Restoring a Saved Configuration for an iSCSI target ......................................... 26  
   5.4 Configuring an iSCSI Initiator ............................................................................ 26  
   5.5 Updating the Discovery Database ...................................................................... 28  
6 Using Multipathing for Efficient Storage ........................................................................ 31  
   6.1 About Device Multipathing ................................................................................ 31  
   6.2 Configuring Multipathing .................................................................................. 32  
   6.3 Working With the Multipathing Configuration File ............................................. 33
Preface

*Oracle® Linux 8: Managing Storage Devices* describes how to configure and manage disk partitions, swap space, logical volumes, software RAID, block device encryption, iSCSI storage, and multipathing.

**Audience**

This document is intended for administrators who need to configure and administer Oracle Linux systems. It is assumed that readers are familiar with web technologies and have a general understanding of using the Linux operating system, including knowledge of how to use a text editor such as *emacs* or *vim*, essential commands such as *cd, chmod, chown, ls, mkdir, mv, ps, pwd, and rm*, and using the *man* command to view manual pages.

**Document Organization**

The document is organized into the following chapters:

- Chapter 1, *Configuring Storage Drives*
- Chapter 2, *Working With Logical Volume Manager*
- Chapter 3, *Working With Software RAID*
- Chapter 4, *Using Encrypted Block Devices*
- Chapter 5, *Working With iSCSI Devices*
- Chapter 6, *Using Multipathing for Efficient Storage*

**Related Documents**

The documentation for this product is available at:

*Oracle® Linux 8 Documentation*

**Conventions**

The following text conventions are used in this document:

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

**Documentation Accessibility**

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

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.
Chapter 1 Configuring Storage Drives

Table of Contents

1.1 Using Disk Partitions ........................................................................................................... 1
  1.1.1 About Disk Partitions ........................................................................................................ 1
  1.1.2 Partitioning Disks by Using fdisk .................................................................................. 2
  1.1.3 Partitioning Disks by Using parted ................................................................................. 4
  1.1.4 Mapping Partition Tables to Devices .............................................................................. 6
1.2 Implementing Swap Spaces ............................................................................................... 7
  1.2.1 About Swap Space .......................................................................................................... 8
  1.2.2 Creating a Swap File ........................................................................................................ 8
  1.2.3 Creating a Swap Partition ............................................................................................... 8
  1.2.4 Viewing Swap Space Usage ........................................................................................... 9
  1.2.5 Removing a Swap File or Swap Partition ....................................................................... 9
1.3 Recommendations for Solid State Drives ........................................................................... 9

This chapter describes how to configure storage drives such as disks and SSDs with partitions and swap spaces for efficient storage and preservation of data.

1.1 Using Disk Partitions

All storage devices, from hard disks to solid state drives to SD cards, must be partitioned in order to become usable. A device must have at least one partition, although you can create multiple partitions on it.

1.1.1 About Disk Partitions

Partitioning means dividing a disk drive into one or more reserved areas called partitions. Information about these partitions are stored on 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. Additionally, two other partitions are typically reserved for swap space and the boot file system. On x86 and x86_64 systems, the system BIOS can 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 GRUB bootloader to access the kernel image and other files that are required to boot the system.

You create additional partitions to simplify backups, enhance system security, and meet other needs, such as setting up development sandboxes and test areas. You can add partitions to store data that frequently changes, such as user home directories, databases, and log file directories.

For hard disks with a master boot record (MBR), the partitioning scheme supports up to 4 primary partitions. In turn, a primary partition can further be divided into up to 11 logical partitions. The primary partition that contains the logical partitions is known as an extended partition. The 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. The GPT partition scheme does not use the concept of extended or logical partitions. If your disk's size is larger than 2 TB, you should use GPT to configure the device's partitions.

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), LUNs on storage arrays, and host RAID adapters.

1.1.2 Partitioning Disks by Using fdisk

To create and manage hard disks that use MBRs, you use the fdisk command. Alternatively, you can use the cfdisk utility, which is a text-based, graphical version of fdisk.

Before running fdisk, complete the following requirements first:

- Unmount any mounted partition on the disk.
- Disable any partition that is being used as swap space by using the swapoff command.
- Backup the data on the disk to be configured.

fdisk can be used either interactively or directly with command-line options and arguments.

**Note**

The two modes can differ in the options they support to perform specific actions. To list supported options while in interactive mode, enter `m` at the mode's prompt. For supported options in the command line mode, type:

```
$ fdisk -h
```

To run fdisk interactively, specify only the name of the disk device as an argument, for example:

```
$ sudo fdisk /dev/sda
```

Welcome to fdisk (util-linux 2.32.1)
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Command (m for help):

The following commands are useful for managing partitions:

- `p` Displays the current partition table.
- `n` Initiates the process for creating new partitions.
- `t` Changes the partition type.

**Tip**

To list all the supported partition types, enter `l`.

- `w` Commits changes you made to the partition table, then exits the interactive session.
- `q` Disregards any configuration changes you made and exits the session.
- `m` Displays all the supported commands in the interactive mode.

For more information, see the cfdisk(8) and fdisk(8) manual pages.
About the Partition Table

When you enter `p` at the prompt, output similar to the following is displayed:

```
Command (m for help): p
Disk /dev/sda: 36.5 GiB, 39191576576 bytes, 76546048 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optional): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x67fb0c7a

Device     Boot   Start      End  Sectors  Size Id Type
/dev/sda1  *       2048  1026047  1024000  500M 83 Linux
/dev/sda2       1026048 76546047 75520000   36G 8e Linux LVM
```

The output contains device information summary such as disk size and disklabel type, as well as partition details. The partition details are specified under the following field names:

- **Device**: Lists the current partitions on the device.
- **Boot**: Identifies the boot partition with an asterisk (*). This partition contains the files that the GRUB bootloader needs to boot the system. Only one partition can be bootable.
- **Start** and **End**: Lists the start and end offsets in sectors that mark a sector's boundaries. All partitions are aligned on one-megabyte boundaries.
- **Sectors**: Displays sector sizes.
- **Size**: Displays partition sizes.
- **Id** and **Type**: Indicates a representative number and its corresponding representative number.

Oracle Linux typically supports the following types:

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Extended</td>
</tr>
<tr>
<td>82</td>
<td>Linux swap</td>
</tr>
<tr>
<td>83</td>
<td>Linux</td>
</tr>
<tr>
<td>8e</td>
<td>Linux LVM</td>
</tr>
</tbody>
</table>

Creating Partitions

The following example demonstrates how to use the different `fdisk` interactive commands to partition a disk. Specifically, 2 partitions are created on `/dev/sdb`. The first partition is assigned 2 GB while the second partition uses all the remaining disk space.

```
$ sudo fdisk /dev/sdb
...
Command (m for help): n
```
1.1.3 Partitioning Disks by Using parted

To create and manage hard disks that use GPTs, you use the \texttt{parted} command. The command enables you to perform typical partition operations as \texttt{fdisk}. However, with its support for a larger set of commands as well as more disk label types including GPT disks, \texttt{parted} is more advanced.

Before running \texttt{parted}, complete the following requirements first:

- Unmount any mounted partition on the disk.
- Disable any partition that is being used as swap space by using the \texttt{swapoff} command.
- Backup the data on the disk to be configured.

You can use \texttt{parted} either interactively or directly with command line arguments. To run \texttt{parted} interactively, specify only the name of the disk device as an argument, for example:

```
$ sudo parted /dev/sdb
GNU Parted 3.2
Using /dev/sdb
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted)
```

The following commands are useful for managing partitions:
Partitioning Disks by Using parted

print Displays the current partition table.
mklabel Creates a partition type according to the label you choose.
mkpart Initiates the process for creating new partitions.
quit Exits the session.

Note
In interactive sessions, changes are committed to disk immediately. Unlike fdisk, the parted utility does not have an option for quitting without saving changes.

help Displays all the supported commands in the interactive mode.

The following example demonstrates how to use the different parted commands to create 2 disk partitions. The first partition is assigned 2 GB while the second partition uses all the remaining disk space.

```bash
$ sudo parted /dev/sdb
GNU Parted 3.2
Using /dev/sdb
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted) print
Model: ATA VBOX HARDDISK (scsi)
Disk /dev/sdb: 16.8GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number Start End Size Type File system Flags
(parted) mkpart
Partition type? primary/extended? primary
File system type? [ext2]? <Enter>
Start? 1
End? 2GB
(parted) print
Model: ATA VBOX HARDDISK (scsi)
Disk /dev/sdb: 16.8GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number Start End Size Type File system Flags
1 1049kB 2000MB 1999MB primary ext2 lba

(parted) mkpart
Partition type? primary/extended? primary
File system type? [ext2]? <Enter>
Start? 2001
End? -0
(parted) print
Model: ATA VBOX HARDDISK (scsi)
Disk /dev/sdb: 16.8GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number Start End Size Type File system Flags
1 1049kB 2000MB 1999MB primary ext2 lba
2 2001MB 16.8GB 14.8GB primary ext2 lba

(parted) quit
```
Note

Unless you specify otherwise, the size for the Start and End offsets is in megabytes. To use another unit of measure, type the value and the unit together, for example, 2GB. To assign all remaining disk space to the partition, enter -0 for the End offset as shown in the example.

By default, parted creates msdos-labeled partitions. When partitioning with this label, you are also prompted for the partition type. Partition types can be primary, extended, or logical.

If you want to use a different label, you would need to specify that label first with the mklabel command before creating the partition. Depending on the label, you would be prompted during the partitioning process for additional information, such as the partition name, as shown in the following example:

```bash
$ sudo parted /dev/sdb
GNU Parted 3.2
Using /dev/sdb
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/sdb will be destroyed and all data on this disk will be lost. Do you want to continue?
Yes/No? yes
(parted) print
Model: ATA VBOX HARDDISK (scsi)
Disk /dev/sdb: 16.8GB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

(parted) mkpart
Partition name? []? Example
File system type? [ext2]? linux-swap
Start? 1
End? 2GB
(parted) print
Model: ATA VBOX HARDDISK (scsi)
Disk /dev/sdb: 16.8GB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number Start End Size File system Name Flags
1 1049kB 2000MB 1999MB linux-swap(v1) Example

(parted) quit
```

To know which types of file systems and labels are supported by parted, consult the GNU Parted User Manual at https://www.gnu.org/software/parted/manual/, or enter info parted to view the online user manual. For additional information, see the parted(8) manual page.

1.1.4 Mapping Partition Tables to Devices

The kpartx utility maps to device files the partitions of any block device or file that contains a partition table. The command reads the partition table, creates device files for the partitions, and stores the device files in /dev/mapper. Each device file represents a disk volume or a disk partition on a device or within an image file.
Implementing Swap Spaces

The –a option creates the device mappings. The following example uses the disk partitions that were created in Creating Partitions as basis for creating the mapping. The example begins by showing the partition table:

```bash
$ sudo fdisk -l /dev/sdb
... Device Boot Start   End   Sectors Size Id Type
/dev/sdb1          2048  3907583  3905536  1.9G 83 Linux
/dev/sdb2         3907584 32767999 28860416 13.8G 83 Linux
$ sudo kpartx -av /dev/sdb
add map sdb1 (253:2): 0 3905536 linear 8:16 2048
add map sdb2 (253:3): 0 28860416 linear 8:16 3907584
$ ls /dev/mapper
control  sdb1  sdb2  vg_main-lv_root  vg_main-lv_swap
```

The –l option lists partitions in the device.

In the following example, the first column of the output identifies the device files in /dev/mapper.

```bash
$ sudo kpartx -l /dev/sdb
sdb1 : 0 3905536 /dev/sdb 2048
sdb2 : 0 28860416 /dev/sdb 3907584
```

The `kpartx` command also works with image files such as an installation image. For example, for an image file `system.img`, you can do the following:

```bash
$ sudo kpartx -a system.img
$ 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
```

The output shows that the drive image contains four partitions.

If a partition contains a file system, you can mount the file system and view its contents, for example:

```bash
# mkdir /mnt/sysimage
# mount /dev/mapper/loop0p1 /mnt/sysimage
# 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
...
# umount /mnt/sysimage
```

The –d option removes the device mappings:

```bash
# kpartx -d system.img
# ls /dev/mapper
control
```

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

### 1.2 Implementing Swap Spaces

Swap spaces are a way by which the operating system manages resources in the system and ensures its performance.
1.2.1 About Swap Space

Oracle Linux uses swap space if your system does not have enough physical memory for ongoing processes. When available memory is low, the operating system writes inactive pages to swap space on the disk, and thus free up physical memory.

However, swap space is not an effective solution to memory shortage. Swap space is located on disk drives, which have much slower access times than physical memory. Writing to swap space effectively degrades system performance. If your system often resorts to swapping, you should add more physical memory, not more swap space.

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

1.2.2 Creating a Swap File

1. Use the `dd` command to create a file of the required size, for example, one million 1KB blocks.

   ```
   $ sudo dd if=/dev/zero of=/swapfile bs=1024 count=1000000
   1000000+0 records in
   1000000+0 records out
   1024000000 bytes (1.0 GB, 977 MiB) copied, 6.10298 s, 168 MB/s
   ```

2. Initialize the file as a swap file.

   ```
   $ sudo mkswap /swapfile
   mkswap: /swapfile: insecure permissions 0644, 0600 suggested.
   Setting up swapspace version 1, size = 976.6 MiB (1023995904 bytes)
   no label, UUID=43964855-e81f-414c-a61c-370408085ba4
   ```

3. Change the permissions on the file so that it is not world readable.

   ```
   $ sudo chmod 0600 /swapfile
   ```

4. Add an entry to the `/etc/fstab` file so that the system uses the swap file at system reboots, for example:

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

5. Regenerate the mount units and register the new configuration in `/etc/fstab`.

   ```
   $ sudo systemctl daemon-reload
   ```

6. Activate the swap file.

   ```
   $ sudo swapon /swapfile
   ```

7. (Optional) Test whether the new swap file was successfully created by inspecting the active swap space:

   ```
   $ cat /proc/swaps
   $ free -h
   ```

1.2.3 Creating a Swap Partition

1. Create the swap partition by using either `fdisk` or `parted`.

   - If using `fdisk`, create the partition as discussed in Creating Partitions. Then use `t` to change the partition type from the default to 82 Linux swap / ".

8
Viewing Swap Space Usage

- If using `parted`, specify `linux-swap` at the `File system type?` prompt as shown in Section 1.1.3, “Partitioning Disks by Using parted”.

2. Initialize the partition as a swap partition.
   
   For example, if the partition is `/dev/sda2`, type:
   
   ```
   $ sudo mkswap /dev/sda2
   ```

3. Enable swapping to the swap partition:
   
   ```
   $ sudo swapon /dev/sda2
   ```

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
   ```

### 1.2.4 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 4GB swap partition on `/dev/sda2` and a 1GB swap file, `/swapfile`. The `Priority` column shows that the operating system to write 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
```

```
# free | grep Swap
Swap:      5128752        388    5128364
```

### 1.2.5 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 no longer need it.

### 1.3 Recommendations for Solid State Drives

Just like other storage devices, solid state drives (SSDs) require their partitions to be on 1 MB boundaries.

For btrfs and ext4 file systems on SSDs, 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. However, the option also has a negative impact on performance, even for SSDs that support queued discards.

Instead, use the `fstrim` command to discard empty and unused blocks, especially before reinstalling the operating system or before creating a new file system on an SSD. Schedule `fstrim` to run when impact on system performance is minimal. You can also apply `fstrim` to a specific range of blocks rather than the whole file system.

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

Btrfs automatically enables SSD optimization for a device if the value of `/sys/block/device/queue/rotational` is 0, such as in the case of Xen Virtual Devices (XVD). If btrfs does not detect a device as being an SSD, enable SSD optimization by specifying the `ssd` option to `mount`. Note, however, that setting the `ssd` option does not imply that `discard` is also set.

To disable SSD optimization, specify the `nossd` option to `mount`.

Using a minimal journal size of 1024 file-system blocks for ext4 on an SSD improves performance. However, journaling also improves the robustness of the file system, and therefore should not be completely disabled.

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. Use the following commands to change the value:
Chapter 2 Working With Logical Volume Manager

Table of Contents

2.1 About Logical Volume Manager ................................................................. 11
2.2 Setting Up Logical Volumes ........................................................................ 11
  2.2.1 Initializing and Managing Physical Volumes ........................................ 12
  2.2.2 Creating and Managing Volume Groups .............................................. 12
  2.2.3 Creating and Managing Logical Volumes ............................................ 13
  2.2.4 Creating Logical Volume Snapshots .................................................... 13
2.3 Using Thinly-Provisioned Logical Volumes ................................................. 14
  2.3.1 Configuring and Managing Thinly-Provisioned Logical Volumes .......... 14
  2.3.2 Using snapper With Thinly-Provisioned Logical Volumes .................... 15

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

2.1 About Logical Volume Manager

Logical Volume Manager (LVM) enables you to manage multiple physical volumes and configure mirroring and striping of logical volumes. Through its use of the device mapper (DM) to create an abstraction layer, LVM provides you the capability to by which you can configure physical and logical volumes. With LVM, you obtain data redundancy as well increased I/O performance.

In LVM, you first create volume groups from physical volumes. Physical volumes are storage devices such as disk array LUNs, software or hardware RAID devices, hard drives, and disk partitions. Over these physical volumes, you create volume groups. In turn, you configure logical volumes in a volume group. Logical volumes become the foundation for configuring software RAID, encryption, and other storage features.

You create file systems on logical volumes and mount the logical volume devices in the same way as you would a physical device. If a file system on a logical volume becomes full with data, you can increase the volume's capacity by using free space in the volume group. You can then grow the file system, if the file system supports that capability. Physical storage devices can be added to a volume group to further increase its capacity.

LVM is non-disruptive and transparent to users. Thus, management tasks such as increasing logical volume sizes, changing their layouts dynamically, or reconfiguring physical volumes do not require any system downtime.

2.2 Setting Up Logical Volumes

To configure logical volumes with LVM, follow this sequence of steps:

1. Backup the data that might exist in the devices designated for the physical volume.

2. Unmount those designated devices.

   If the devices are mounted, creating physical volumes fails.

3. Create physical volumes from selected storage devices.

   $ sudo pvcreate [options] devices
4. Create a volume group from physical volumes.

```bash
sudo vgcreate [options] vol_group physical_vols
```

5. Configure logical volumes over the volume group.

```bash
sudo lvcreate [options] --size size --name logical_vol vol_group
```

6. As needed, create snapshots of logical volumes.

```bash
sudo lvcreate --snapshot --size size --name snapshot_name logical_vol
```

The next sections describe the procedure in further detail and provide examples to implement each step.

### 2.2.1 Initializing and Managing Physical Volumes

The following example sets up /dev/sdb, /dev/sdc, /dev/sdd, and /dev/sde as physical volumes:

```bash
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, use the `pvdisplay`, `pvs`, and `pvscan` commands.

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

```bash
sudo pvremove device
```

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

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

### 2.2.2 Creating and Managing Volume Groups

The following example creates the volume group `myvg` from the newly created physical volumes:

```bash
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. An extent, with a default size of 4 MB, is the smallest unit that LVM uses when allocating storage to logical volumes.

The *allocation policy* determines how LVM allocates extents from either a volume group or a logical volume. The default allocation policy for a volume group is *normal*, whose rules include, for example, not placing parallel stripes on the same physical volume. For a logical volume, the default allocation policy is *inherit*, which means that the logical volume uses the same policy as the volume group. Other allocation policies are *anywhere*, *contiguous* and *cling*, and *cling_by_tags*.

To change allocation policies, use the `lvchange` or `vgchange` commands. As an alternative, set the allocation policy of your choice directly when creating a volume group or logical volume.
Creating and Managing Logical Volumes

The `vgextend` and `vgreduce` commands respectively adds physical volumes to a volume group or removes them. The commands enable you to manipulate the size of the volume group.

```
$ sudo vgextend | vgreduce [options] vol_group physical_vol
```

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

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

```
$ sudo vgremove vol_group
```

The command warns you if logical volumes exist in the group and prompts for confirmation.

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.

### 2.2.3 Creating and Managing Logical Volumes

This example creates the logical volume `mylv` of size 2 GB in the volume group `myvg`:

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

Archiving volume group “myvg” metadata (seqno 1).
Creating logical volume mylv
Create volume group backup “/etc/lvm/backup/myvg” (seqno 2).
Activating logical volume myvg/mylv.
...
Logical volume “mylv” created.

`lvcreate` uses the device mapper to create a block device file entry under `/dev` for each logical volume. The command also 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`, to which `/dev/mapper/myvg-mylv` and `/dev/myvg/mylv` are symbolically linked.

In commands or scripts, always refer to the devices in `/dev/mapper` or `/dev/volume_group`, rather than to `/dev/dm-*`. Those names are persistent and are created automatically by the device mapper early in the boot process. In contrast, the `/dev/dm-*` devices are not guaranteed to be persistent across reboots.

You manage and use a logical volume as you would a physical storage device, such as configuring a logical volume as a file system, a swap partition, an Automatic Storage Management (ASM) disk, or a raw device.

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

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

```
$ sudo lvremove vol_group/logical_vol
```

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.

### 2.2.4 Creating Logical Volume Snapshots

To create a snapshot of an existing logical volume, use `lvcreate --snapshot`, for example:

```
13
```
You can mount and modify the contents of the snapshot independently of the original volume. Or, you can preserve the snapshot as a record of the state of the original volume at the time that the snapshot was taken.

The snapshot usually occupies less space than the original volume, depending on how much the contents of the volumes diverge over time. In the example, assume that the snapshot only requires one quarter of the space of the original volume. To calculate how much data is allocated to the snapshot, do the following:

1. Issue the `lvs` command.
2. From the command output, check the value under the `Snap%` column.
   - A value approaching 100% indicates that the snapshot is low on storage space.
3. Use `lvresize` to either grow the snapshot or reduce its size to save storage space.

To merge a snapshot with its original volume, use the `lvconvert --merge` command.

To remove a logical volume snapshot from a volume group, use the `lvremove` command as you would for a logical volume, for example:

```bash
sudo lvremove myvg/mlv-snapshot
```

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

### 2.3 Using 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 called 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.

#### 2.3.1 Configuring and Managing Thinly-Provisioned Logical Volumes

Creating thinly-provisioned logical volumes involves two steps:

1. Create a thin pool.
   ```bash
   sudo lvcreate --size size --thin vol_group/pool_name
   ```

2. Create a thinly-provisioned logical volume.
   ```bash
   sudo lvcreate --virtualsize size --thin vol_group/thin_pool_name --name logical_vol
   ```

In the following example, the thin pool `mytp` of size 1 GB is first created from the volume group `myvg`. Then, the thinly-provisioned logical volume `mytv` is created with a virtual size of 2 GB.

```bash
sudo lvcreate --size 1g --thin myvg/mytp
Logical volume "mytp" created

sudo lvcreate --virtualsize 2g --thin myvg/mytp --name mytv
Logical volume "mytv" created
```

Note that the size of `mytp` is less than that of `mytv`. 

To create a snapshot of `mytv`, do not specify the size of the snapshot. Otherwise, its storage would not be provisioned from `mytp`.

```
$ sudo lvcreate --snapshot --name mytv-snapshot myvg/mytv
Logical volume "mytv-snapshot" created
```

If the volume group has sufficient space, use the `lvresize` command as needed 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 more information, see the `lvcreate(8)` and `lvresize(8)` manual pages.

### 2.3.2 Using snapper With Thinly-Provisioned Logical Volumes

The `snapper` utility is another tool for creating and maintaining 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
```

<table>
<thead>
<tr>
<th>config_name</th>
<th>Name of the configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>fs_type</td>
<td>File system type (ext4 or xfs)</td>
</tr>
<tr>
<td>fs_name</td>
<td>Path of the file system.</td>
</tr>
</tbody>
</table>

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.

With `snapper`, you can create 3 types of snapshots:

**post**

A *post snapshot* records 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**

A *pre snapshot* records the state of a volume immediately 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**

A single snapshot records the state of a volume but does not have any association with other snapshots of the volume.

For example, the following commands create a pre snapshot and a 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 .. ..
```

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'
```

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.
3.1 About Software RAID

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

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

- **Linear RAID (spanning)**: Combines drives as a larger virtual drive. This level provides no data redundancy nor 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 as a consequence of the need to calculate parity information and to write the 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 repopulate the new drive 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.
Creating Software RAID Devices

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 remains available. RAID 0+1 is usually as expensive as or slightly more expensive than RAID-1.

3.2 Creating Software RAID Devices

1. Run the `mdadm` command to create the MD RAID device as follows:

   ```
   # mdadm --create md_device --level=RAID_level [options] --raid-devices=N devices
   
   md_device               Name of the RAID device, for example, /dev/md0.
   RAID_level              Level number of the RAID to create, for example, 5 for a RAID-5 configuration.
   --raid-devices=N         Number of devices to become part of the RAID configuration.
   devices                 Devices to be configured as RAID, for example, /dev/sd[bcd] for 3 devices for the RAID configuration.
   
   The devices you list must total to the number you specified for --raid-devices.
   
   This example creates 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]
   
   This example creates a RAID-5 device /dev/md1 out of 4 devices. One device is configured as a spare for expansion, reconfiguration, or replacement of failed drives:
   
   $ sudo mdadm --create /dev/md1 --level=5 --raid-devices=3 --spare-devices=1 /dev/sd[bcde]
   ```

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

   ```
   $ sudo mdadm --examine --scan >> /etc/mdadm.conf
   
   Based on the configuration file, `mdadm` assembles the arrays at boot time.
   
   For example, the following entries 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`
   ```

An MD RAID device is used in the same way as any physical storage device. For example, the RAID device can be configured as an LVM physical volume, a file system, a swap partition, an Automatic Storage Management (ASM) disk, or a raw device.
Creating Software RAID Devices

To check the status of the MD RAID devices, view `/proc/mdstat`:

```
$ cat /proc/mdstat
Personalities: [raid1]
md0 : active raid1 sdg[1] sdf[0]
```

To display a summary or detailed information about MD RAID devices, use the `--query` or `--detail` option, respectively, with `mdadm`.

For more information, see the `md(4)`, `mdadm(8)`, and `mdadm.conf(5)` manual pages.
Chapter 4 Using Encrypted Block Devices

Table of Contents

4.1 About Encrypted Block Devices ................................................................. 21
4.2 Creating Encrypted Volumes ................................................................... 21

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

4.1 About Encrypted Block Devices

The device mapper supports the encryption of block devices through the dm-crypt device driver. Data on these devices are accessible at boot time only with proper credentials. dm-crypt encrypts disk partitions, RAID volumes, and LVM physical volumes, regardless of their contents.

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

4.2 Creating Encrypted Volumes

The cryptsetup utility sets up Linux Unified Key Setup (LUKS) encryption on the device and to manage authentication.

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

   $ sudo cryptsetup luksFormat /dev/sdd

   WARNING!

   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, for example:

   $ sudo cryptsetup luksOpen /dev/sdd cryptfs

   Enter passphrase for /dev/sdd: passphrase

   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 for the passphrase at boot time.

You use an encrypted volume in the same way as you would a physical storage device, for example, as an LVM physical volume, file system, swap partition, Automatic Storage Management (ASM) disk, or raw device. For example, to mount the encrypted volume automatically, 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:

   $ sudo cryptsetup status cryptfs
Creating Encrypted Volumes

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

To remove the device mapping:

1. Unmount any existing file system in the encrypted volume.
2. Remove the mapped device from /dev/mapper.

   For example, for the encrypted volume cryptfs, type:

   $ sudo cryptsetup luksClose /dev/mapper/cryptfs

For more information, see the cryptsetup(8) and crypttab(5) man pages.
Chapter 5 Working With iSCSI Devices

Table of Contents

5.1 About Linux-IO Storage Configuration ............................................................... 23
5.2 Configuring an iSCSI Target ............................................................................. 24
5.3 Restoring a Saved Configuration for an iSCSI target ..................................... 26
5.4 Configuring an iSCSI Initiator ......................................................................... 26
5.5 Updating the Discovery Database ................................................................. 28

This chapter discusses using iSCSI devices for data storage.

5.1 About Linux-IO Storage Configuration

Oracle Linux with both Unbreakable Enterprise Kernel 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 manage LIO by using the `targetcli` shell provided in the `targetcli` package.

Fibre Channel over Ethernet (FCoE) encapsulates Fibre Channel packets in Ethernet frames, which enables them to be sent over Ethernet networks. To configure FCoE storage, you need to install the `fcoe-utils` package that includes both 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 to support data transfer over long distances and sharing of storage by client systems. iSCSI uses the existing IP infrastructure and 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. The iSCSI initiator functionality is not available at the level of the system BIOS. Thus, 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.

### 5.2 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.

In the example, the `targetcli` command saves the current configuration to `/etc/target/saveconfig.json`. See the `targetcli(8)` manual page for additional information.

1. Run the `targetcli` interactive shell:

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

2. (Optional): 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]
   ```

3. 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.

4. 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.
   ```

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

   ```
   /iscsi> ls
   o- iscsi .................................................... [Targets: 1]
   ```
6. Change to the `luns` subdirectory of the TPG directory hierarchy and add the LUNs to the target portal group:

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

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

```bash
/iscsi/iqn.2013-01.com.mydom.host01.x8664:sn.ef8e14f87344/tpg1/luns> cd ../portals
/iscsi/iqn.2013-01.com.mydom.host01.x8664:sn.ef8e14f87344/tpg1/portals>
/iscsi/iqn.2013-01.com.mydom.host01.x8664:sn.ef8e14f87344/tpg1/portals> create 10.150.30.72 3260
Using default IP port 3260
Created network portal 10.150.30.72:3260.
```

The default TCP port number is **3260**.

---

**Note**

An existing default portal would cause the portal creation to fail and a message similar to the following is generated:

```
Could not create NetworkPortal in configFS
```

To resolve the issue, delete the default portal, then create the new portal again, for example:

```bash
/iscsi/iqn.2013-01.com.mydom.host01.x8664:sn.ef8e14f87344/tpg1/portals> delete 0.0.0.0 ip_port=3260
```

8. Enable TCP port 3260 either by adding the port or adding the iSCSI target:

- Adding the port:
  ```bash
  $ sudo firewall-cmd --permanent --add-port=3260/tcp
  ```

- Adding the target:
  ```bash
  $ sudo firewall-cmd --permanent --add-service iqan.2013-01.com.mydom.host01.x8664:sn.ef8e14f87344
  --add-service 10.150.30.72
  --add-service 3260
  ```

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

```bash
/iscsi/iqn.2013-01.com.mydom.host01.x8664:sn.ef8e14f87344/tpg1/portals> ls /
```

```
io- backstores .................................................. [Storage Objects: 0]
io- block ...................................................... [Storage Objects: 1]
io- LUN_0 ...................................................... [/dev/sdb (10.0GiB) write-thru activated]
io- LUN_1 ...................................................... [/dev/sdc (10.0GiB) write-thru activated]
io- fileio ...................................................... [Storage Objects: 0]
io- pscsi ...................................................... [Storage Objects: 0]
io- ramdisk .................................................... [Storage Objects: 0]
io- iscsi ........................................................ [Targets: 1]
io- iqan.2013-01.com.mydom.host01.x8664:sn.ef8e14f87344 ............ [TPGs: 1]
io- tpg1 ...................................................... [no-gen-acls, no-auth]
io- acls ....................................................... [ACLs: 0]
io- luns ....................................................... [LUNs: 1]
```
10. Configure the access rights for logins by initiators.

For example, to configure a demonstration mode that does not require authentication, change to the TGP directory and set the attributes as shown in the following example:

```
/iscsi/iqn.20.../tpg1/portals> cd ..
/iscsi/iqn.20.../14f87344/tpg1>
set attribute authentication=0 demo_mode_write_protect=0
genenerate_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**

The demonstration mode is inherently insecure. For information about configuring secure authentication modes, see [http://linux-iscsi.org/wiki/ISCSI#Define_access_rights](http://linux-iscsi.org/wiki/ISCSI#Define_access_rights).

11. Change to the root (/) directory and save the configuration.

This step ensures that the changes persist across system reboots. Omitting the step might result in an empty configuration.

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

12. Enable the target service.

```
$ sudo systemctl enable target.service
```

### 5.3 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'.
```

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

The `/etc/target/saveconfig.json` file stores the most recently saved configuration.

As an alternative, run the following command to restore saved configurations from previous versions:

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

### 5.4 Configuring an iSCSI Initiator

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

```
$ sudo dnf install iscsi-initiator-utils
```
2. Use a discovery method, such as SendTargets or the Internet Storage Name Service (iSNS), to discover the iSCSI targets at the specified IP address.

   For example, you would use SendTargets as follows:

   
   ```bash
   $ 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
   ```

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

   **Note**

   Before running the discovery process, ensure that the firewall is configured to allow communication with an iSCSI target and that ICMP traffic is permitted.

3. Display information about the targets that are now stored in the discovery database.

   ```bash
   $ 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:

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

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

   ```bash
   $ 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 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
   ```
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/
lwxrwxrwx 1 root root 9 May 15 21:05
  ...
lwxrwxrwx 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, for example:

```
$ grep sdb /var/log/messages
... 
May 18 14:19:36 localhost kernel: [12079.963376] sd 8:0:0:0: [sdb] Attached SCSI disk ...
```

You 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.

When creating mount entries for iSCSI LUNs in /etc/fstab, specify the _netdev option, for example:

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

This option indicates that 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
When adding iSCSI LUN entries to /etc/fstab, refer to the LUN by using UUID=UUID rather than the device path. A device path can change after reconnecting the storage or rebooting the system. To display the UUID of a block device, the blkid command.

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.

### 5.5 Updating the Discovery Database

If the LUNs that are available on an iSCSI target change, use the iscsiadm command on an iSCSI initiator to update the entries in its discovery database. The following example assumes that the target supports the SendTargets discovery method.
Updating the Discovery Database

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

```
$ sudo iscsiadm --mode discoverydb -t 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.
Chapter 6 Using Multipathing for Efficient Storage

Table of Contents

6.1 About Device Multipathing ................................................................. 31
6.2 Configuring Multipathing .................................................................... 32
6.3 Working With the Multipathing Configuration File ............................... 33

This chapter describes how to use multipathing to implement redundancy and failover.

6.1 About Device Multipathing

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 to represent multiple I/O paths between a server and a storage device as a single path.

You would typically configure multipathing on a system that can access storage on a Fibre Channel-based storage area network (SAN), or 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 both paths connect 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.
Configuring Multipathing

The multipathing service (multipathd) handles 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.

The naming of multipath devices is managed by multipathing’s user_friendly_names property in the multipath.conf file. If set to no, then the devices are named based on their World Wide Identifiers (WWIDs), that is, /dev/mapper/WWID. WWIDs are unique to their respective devices.

To obtain the WWID of a SCSI device, use the scsi_id command:

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

If the property is set to yes, the devices are mapped as /dev/mapper/mpathN, where N is the multipath group number. In addition, you can use the alias attribute to assign meaningful names to the devices. See Section 6.3, “Working With the Multipathing Configuration File”.

To check the status of user_friendly_names as well as other DM-multipath settings, issue the mpathconf command, for example:

```
$ sudo mpathconf
multipath is enabled
find_multipaths is enabled
user_friendly_names is enabled
dm_multipath modules is loaded
multipathd is running
```

Alternatively, you can view the settings in /etc/multipath.conf.

You can use the multipath device in /dev/mapper to reference the storage in the same way as you would any other 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.

6.2 Configuring Multipathing

1. Install the device-mapper-multipath package.

```
$ sudo dnf 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 know the status of multipathing, type:

```
$ sudo mpathconf
```
4. Edit `/etc/multipath.conf` as required.

For details, see Section 6.3, “Working With the Multipathing Configuration File”

To display the current multipath configuration, specify the `-ll` option to the `multipath` command, for example:

```
$ sudo multipath -ll
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 sample 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 round-robin I/O path selection. The WWID that identifies the storage is `360000970000292602744533030303730` 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.

### 6.3 Working With The Multipathing Configuration File

Through the `/etc/multipath.conf` file, you can add a combination of definitions that customizes multipathing according to your system environment setup, such as in the following example:

```plaintext
defaults {
  udev_dir /dev
  polling_interval 10
  path_selector "round-robin 0"
  path_grouping_policy multibus
  getuid_callout "/lib/udev/scsi_id --whitelisted --device=/dev/%n"
  prio alua
  path_checker readsector0
  rr_min_io 100
  max_fds 8192
  rr_weight priorities
  failback immediate
  no_path_retry fail
  user_friendly_names yes
}

blacklist {
  # Blacklist by WWID
  wwid "**"

  # Blacklist by device name
  devnode "^[a-zA-Z0-9]{1,15}$"

  # Blacklist by device type
  device {
    vendor "COMPAQ"
    product "HSV110 (C)COMPAQ"
  }
}

blacklist_exceptions {
  wwid "360000970000292602744533030303730"
  wwid "36000097000029260274453303032443941"
}
```
The file is divided into the following sections:

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

**blacklist**
Defines devices that are excluded from multipath topology discovery. Blacklisted devices cannot be 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.

**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.

For information about the storage arrays that DM-Multipath supports, as well as their default configuration values, see the /
usr/share/doc/device-mapper-multipath-version/multipath.conf.defaults file. The settings in this file can be the basis for entries in /etc/multipath.conf.

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 /etc/multipath.conf would be appropriate for setting up active/passive multipathing to an iSCSI LUN with the specified WWID.

```plaintext
defaults {
    user_friendly_names    yes
    getuid_callout         "/bin/scsi_id --whitelisted --replace-whitespace --device=/dev/%n"
}
multipaths {
    multipath {
        wwid 360000970000292602744533030303730
    }
}
```

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

**Note**

If you edit /etc/multipath.conf, restart the multipathd service to make it re-read the file:

```
$ sudo systemctl restart multipathd
```

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