Oracle Linux 10 Managing Storage Devices





Oracle Linux 10 Managing Storage Devices,

G32910-01

Copyright © 2022, 2025, Oracle and/or its affiliates.

Contents

P	re	fa	CP
		ıa	しし

Documentation License	Vi
Conventions	Vi
Documentation Accessibility	Vi
Access to Oracle Support for Accessibility	Vi
Diversity and Inclusion	Vİ
Using Disk Partitions	
Disk Partitions in Oracle Linux	1-1
Partitioning Disks by Using fdisk	1-1
Displaying the Partition Table	1-3
Creating Partitions	1-4
Partitioning Disks by Using parted	1-4
Creating Partitions	1-5
Customizing Labels	1-6
Automatic Device Mappings for Partitions and File Systems	1-7
Listing Device Mapping Information	1-8
Manually Mapping Partition Tables to Devices	1-10
Creating Device Mappings by Using kpartx	1-10
Listing Partition Mappings For a Device by Using kpartx	1-10
Removing Partition Mappings by Using kpartx	1-11
Implementing Swap Spaces	
Creating a Swap File	2-1
Creating a Swap Partition	2-2
Viewing Swap Space Usage	2-2
Removing a Swap File or Swap Partition	2-2
Recommendations for Solid State Drives	



Working With Logical Volume Manager 4 Initializing and Managing Physical Volumes 4-1 4-2 Creating and Managing Volume Groups 4-3 Creating and Managing Logical Volumes Creating and Managing Grouping With Tags 4-3 Managing Activation and Automatic Activation 4-4 Creating Logical Volume Snapshots 4-6 Using Thinly-Provisioned Logical Volumes 4-7 Configuring and Managing Thinly-Provisioned Logical Volumes 4-7 Using snapper With Thinly-Provisioned Logical Volumes 4-7 5 Working With Software RAID Software RAID Levels 5-1 Creating Software RAID Devices using mdadm 5-2 Creating and Managing Software RAID Devices using LVM 5-3 RAID Level 0 (Striping) LVM Examples 5-4 RAID Level 1 (Mirroring) LVM Examples 5-5 RAID Level 5 (Striping with Distributed Parity) LVM Examples 5-7 RAID Level 6 (Striping with Double Distributed Parity) LVM Examples 5-8 RAID Level 10 (Striping of Mirrored Disks) LVM Examples 5-11 6 Using Encrypted Block Devices About Encrypted Block Devices 6-1 Creating Encrypted Volumes 6-1 7 Working With Linux I-O Storage About iSCSI Devices 7-1 Configuring an iSCSI Target 7-2 Restoring a Saved Configuration for an iSCSI target 7-4 Configuring an iSCSI Initiator 7-5 Updating the Discovery Database 7-7 8 Working With NVMe Storage Over Fibre Channel About NVMe/FC Devices 8-1 8-2 NVMe/FC Storage Adapter Compatibility Install NVMe-CLI and Identify Host Namespace 8-2

Discovering and Connecting to Targets from Broadcom Hosts



8-2

	Discovering and Connecting to Targets from Qlogic Hosts	8-5
9	Using Multipathing for Efficient Storage	
	Device Multipathing Sample Setup	9-1
	Configuring Multipathing	9-3
	Working With the Multipathing Configuration File	9-4



Preface

provides information about storage device management, as well as instructions on how to configure and manage disk partitions, swap space, logical volumes, software RAID, block device encryption, iSCSI storage, NVMe/FC storage, and multipathing.

Documentation License

The content in this document is licensed under the Creative Commons Attribution—Share Alike 4.0 (CC-BY-SA) license. In accordance with CC-BY-SA, if you distribute this content or an adaptation of it, you must provide attribution to Oracle and retain the original copyright notices.

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/.

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

Using Disk Partitions

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

Partitioning divides a disk drive into one or more reserved areas called *partitions*. Information about these partitions are stored in the partition table on the disk drive. The OS treats each partition as a separate disk that can contain a file system.

You create more 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.

Disk Partitions in Oracle Linux

Oracle Linux requires one partition for the root file system. Further, 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.

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 doesn't use the concept of extended or logical partitions. If the disk's size is larger than 2 TB, use GPT to configure the device's partitions.

Note:

When you partition a block storage device, align the 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 would 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, solid state drives (SSDs), LUNs on storage arrays, and host RAID adapters.

Partitioning Disks by Using fdisk

To create and manage hard disks that use MBRs, use the fdisk command. Or, 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's being used as swap space by using the swapoff command.
- Backup the data on the disk to be configured.

 $\verb|fdisk| can be used either interactively or directly with command line options and arguments.\\$



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

```
sudo fdisk /dev/sda

Welcome to fdisk (util-linux 2.40.2)
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:

р

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 1.

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.



Displaying the Partition Table

To display the partition table, enter p at the fdisk prompt, for example:

```
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/optimal): 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

Command (m for help):

The output contains device information summary such as disk size, disklabel type, and 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:

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's not managed by LVM. This is the default partition type.

8e Linux LVM

Linux partition that's managed by LVM.



Creating Partitions

The following example shows how to use the different fdisk interactive commands to partition a disk. 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
```

The command runs a menu-based system where you must select the appropriate responses to configure the partition. Example inputs are displayed in the following interactive session:

```
Command (m for help): n
Partition type
   p primary (0 primary, 0 extended, 4 free)
   e extended (container for logical partitions)
Select (default p): p
Partition number (1-4, default 1): 1
First sector (2048-32767999, default 2048): <Enter>
Last sector, +sectors or +size{K,M,G,T,P} (2048-32767999, default 32767999): +2G
Created a new partition 1 of type 'Linux' and of size 2 GiB.
Command (m for help): n
Partition type
   p primary (1 primary, 0 extended, 3 free)
      extended (container for logical partitions)
Select (default p): p
Partition number (2-4, default 2): 2
First sector (4196352-32767999, default 4196352): <Enter>
Last sector, +sectors or +size{K,M,G,T,P} (4196352-32767999, default 32767999): <Enter>
Created a new partition 2 of type 'Linux' and of size 13.6 GiB.
Command (m for help): p
Disk /dev/sdb: 15.6 GiB, 16777216000 bytes, 32768000 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
Disklabel type: dos
Disk identifier: 0x460247f0
Device Boot Start
                           End Sectors Size Id Type
/dev/sdb1 2048 4196351 4194304 2G 83 Linux
/dev/sdb2 4196352 32767999 28571648 13.6G 83 Linux
Command (m for help): W
The partition table has been altered!
Calling ioctl() to re-read partition table.
Syncing disks.
```

Partitioning Disks by Using parted

To create and manage hard disks that use GPTs, use the parted command. The command enables you to perform typical partition operations as fdisk. However, parted is more

advanced because it supports a larger set of commands and more disk label types including GPT disks.

Before running parted, complete the following requirements first:

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

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

```
sudo parted /dev/sdb

GNU Parted 3.6
Using /dev/sdb

Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted)
```

The following commands are useful for managing partitions:

print

Displays the current partition table.

mklabel

Creates a partition type according to the label you choose.

mkpart

Starts the process for creating new partitions.

quit

Exits the session.



In interactive sessions, changes are committed to disk immediately. Unlike fdisk, the parted utility doesn't have an option for quitting without saving changes.

help

Displays all the supported commands in the interactive mode.

Creating Partitions

The following example shows 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.

```
sudo parted /dev/sdb
```

The command runs a menu-based system where you must select the appropriate responses to configure the partition. Example inputs are displayed in the following interactive session:

```
GNU Parted 3.6
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
      1049kB 2000MB 1999MB primary ext2
(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
       1049kB 2000MB 1999MB primary ext2
       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.

Customizing Labels

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

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 information, such as the partition name, as shown in the following example:

```
sudo parted /dev/sdb
```

The command runs a menu-based system where you must select the appropriate responses to configure the partition. Example inputs are displayed in the following interactive session:

```
GNU Parted 3.6
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:
Number Start End Size File system Name 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 swap
(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 more information, see the parted (8) manual page.

Automatic Device Mappings for Partitions and File Systems

Device mappings are handled automatically on Oracle Linux, by the kernel and the udev service. In the case of disk devices, the kernel automatically creates device mappings for disks and their partitions using a non persistent naming scheme in the form of <code>/dev/sdxy</code>, for example <code>/dev/sda1</code>. A problem with this approach is that the device naming scheme depends on the order in which devices are detected, which means that device names can change easily. Typically, device names change when changes to the boot process exist or when storage devices or their related controllers fail.

Avoid using the non persistent disk or partition names when configuring mount points in /etc/fstab.

Udev is a subsystem that works with the kernel to monitor hardware or device changes and manages events related to changes. Storage devices, partitions, and file systems are all allocated unique identifiers that the udev subsystem can read and use to automatically configure device mappings that you can use to identify the device or partition that you intend to work use. Device mappings for storage devices managed by udev are stored in /dev/disks and you can identify a device by various identifiers, including the unique partition UUID, file system UUID, or the partition label. For instructions about obtaining UUID information, see Listing Device Mapping Information.

When configuring mount points in /etc/fstab, use the file system UUID or, if set, the partition label, for example:

Note that because file systems can span multiple devices, defining mount points against the file system UUID is preferable to using a partition UUID or label. File system UUIDs are assigned when the file system is created and is stored as part of the file system itself. If you copy the file system to a different device, the file system retains the same file system UUID. However, if you reformat a device by using the mkfs command, the device loses the file system UUID and a new UUID is generated.

Listing Device Mapping Information

You can use the <code>lsblk</code> command to list device information for any block device attached to the system:



Tip:

Use the $-\circ$ +UUID option to display the UUIDs for each device and partition listed, or use the -f option to get a display of important file system information.

lsblk -f

Output might appear as follows, indicating a tree of file system mappings:

```
FSTYPE
                             FSVER
                                     LABEL UUID
FSAVAIL FSUSE% MOUNTPOINTS
sda
-sda1
                vfat
                             FAT16
                                           DDD4-C455
94.5M
       5% /boot/efi
                                           8980b45b-a2ce-4df6-93d8-d1e72f3664a0
-sda2
1.5G 26% /boot
∟sda3
               LVM2 member LVM2 001
                                           LzsEPR-Mnbk-kQZY-eV3n-9u1T-1FXZ-
x7ANgD
 -ocivolume-root xfs
                                           26029274-0a04-4dc5-b794-2576b9a16884
28.6G 25% /
 Locivolume-oled xfs
                                           3cbc8301-6f0e-4947-bbe6-f3669b9e6985
       1% /var/oled
```

You can also use the udevadm info command to obtain information about any udev mappings on the system. For example:

```
sudo udevadm info /dev/sda3
```



Output might appear as follows, listing all the device links that udev has created for the device and any other information that udev has on the device:

```
P: /devices/pci0000:00/0000:00:04.0/virtio1/host2/target2:0:0/2:0:0:1/block/sda/sda3
N: sda3
L: 0
S: disk/by-partuuid/18b918a1-16ca-4a07-91c6-455c6dc59fac
S: oracleoci/oraclevda3
S: disk/by-id/wwn-0x60170f5736a64bd7accb6a5e66fe70ee-part3
S: disk/by-path/pci-0000:00:04.0-scsi-0:0:0:1-part3
S: disk/by-id/scsi-360170f5736a64bd7accb6a5e66fe70ee-part3
S: disk/by-id/lvm-pv-uuid-LzsEPR-Mnbk-kQZY-eV3n-9u1T-1FXZ-x7ANgD
E: DEVPATH=/devices/pci0000:00/0000:04.0/virtio1/host2/target2:0:0/2:0:0:1/block/sda/
sda3
E: DEVNAME=/dev/sda3
E: DEVTYPE=partition
E: DISKSEQ=9
E: PARTN=3
E: MAJOR=8
E. MINOR=3
E: SUBSYSTEM=block
E: USEC INITIALIZED=20248855
E: ID SCSI=1
E: ID VENDOR=ORACLE
E: ID VENDOR ENC=ORACLE\x20\x20
E: ID MODEL=BlockVolume
E: ID MODEL ENC=BlockVolume\x20\x20\x20\x20\x20
E: ID REVISION=1.0
E: ID TYPE=disk
E: ID SERIAL=360170f5736a64bd7accb6a5e66fe70ee
E: ID SERIAL SHORT=60170f5736a64bd7accb6a5e66fe70ee
E: ID WWN=0x60170f5736a64bd7
E: ID WWN VENDOR EXTENSION=0xaccb6a5e66fe70ee
E: ID WWN WITH EXTENSION=0x60170f5736a64bd7accb6a5e66fe70ee
E: ID BUS=scsi
E: ID PATH=pci-0000:00:04.0-scsi-0:0:0:1
E: ID PATH TAG=pci-0000 00 04 0-scsi-0 0 0 1
E: ID PART TABLE UUID=a0b1f7d8-e84b-461f-a016-c3fcfed369c3
E: ID PART TABLE TYPE=gpt
E: ID SCSI INQUIRY=1
E: ID FS UUID=LzsEPR-Mnbk-kQZY-eV3n-9u1T-1FXZ-x7ANgD
E: ID FS UUID ENC=LzsEPR-Mnbk-kQZY-eV3n-9u1T-1FXZ-x7ANgD
E: ID FS VERSION=LVM2 001
E: ID FS TYPE=LVM2 member
E: ID FS USAGE=raid
E: ID PART ENTRY SCHEME=gpt
E: ID PART ENTRY UUID=18b918a1-16ca-4a07-91c6-455c6dc59fac
E: ID PART ENTRY TYPE=e6d6d379-f507-44c2-a23c-238f2a3df928
E: ID PART ENTRY NUMBER=3
E: ID_PART_ENTRY_OFFSET=4401152
E: ID PART ENTRY SIZE=100456415
E: ID PART ENTRY DISK=8:0
E: SCSI TPGS=0
E: SCSI TYPE=disk
E: SCSI VENDOR=ORACLE
E: SCSI VENDOR ENC=ORACLE\x20\x20
E: SCSI MODEL=BlockVolume
E: SCSI MODEL ENC=BlockVolume\x20\x20\x20\x20\x20
E: SCSI REVISION=1.0
E: SCSI IDENT LUN NAA REGEXT=60170f5736a64bd7accb6a5e66fe70ee
E: UDISKS IGNORE=1
E: DEVLINKS=/dev/disk/by-partuuid/18b918a1-16ca-4a07-91c6-455c6dc59fac
```

```
/dev/oracleoci/oraclevda3
/dev/disk/by-id/wwn-0x60170f5736a64bd7accb6a5e66fe70ee-part3
/dev/disk/by-path/pci-0000:00:04.0-scsi-0:0:1-part3
/dev/disk/by-id/scsi-360170f5736a64bd7accb6a5e66fe70ee-part3
/dev/disk/by-id/lvm-pv-uuid-LzsEPR-Mnbk-kQZY-eV3n-9u1T-1FXZ-x7ANgD
E: TAGS=:systemd:
E: CURRENT TAGS=:systemd:
```

Manually 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.

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

Creating Device Mappings by Using kpartx

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:

1. Display the partition table.

Map the partitions.

```
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
```

Display /dev/mapper contents.

```
ls /dev/mapper
control sdb1 sdb2 vg_main-lv_root vg_main-lv_swap
```

Listing Partition Mappings For a Device by Using kpartx

To list the partitions in the device, use the -1 option.

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

```
sudo kpartx -1 /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.imq, you can do the following:

```
sudo kpartx -a system.img
sudo kpartx -1 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 of the previous command shows that the drive image contains four partitions.

Removing Partition Mappings by Using kpartx

If a partition isn't in use you can remove the device mapping for the partition by using the -d option:

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



Implementing Swap Spaces

Swap spaces are a way by which the operating system manages resources in the system to ensure efficient performance.

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.

Creating a Swap File

 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, fix with: chmod 0600 /swapfile
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
sudo free -h
```

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 / So.
 - If using parted, specify linux-swap at the File system type? prompt as shown in Partitioning Disks by Using parted.
- 2. Initialize the partition as a swap partition.

For example, if the partition is /dev/sda2, use the following command:

```
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, for example:

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

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 OS writes 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
```

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.



Recommendations for Solid State Drives

Similar to 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.



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 disabled completely.

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 doesn't detect a device as being an SSD, enable SSD optimization by specifying the ssd option to mount. Note, however, that setting the ssd option doesn't imply that discard is also set.

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

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:

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

For additional swap-related information in connection with the btrfs file system, see Creating Swap Files on a Btrfs File System in .

Working With 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 configure physical and logical volumes. With LVM, you obtain data redundancy and 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. Within these volume groups, you configure logical volumes. Logical volumes become the foundation for configuring software RAID, encryption, and other storage features.

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, you can increase the volume's capacity by using free space in the volume group. You can grow a file system if the file system supports that capability. You can add physical storage devices 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 don't require any system down time.

Before setting up logical volumes on the system, complete the following requirements:

- Backup the data on the devices assigned for the physical volume.
- Unmount those devices because you can't create a logical volume on a mounted device.

Configuring logical volumes with LVM involves the following tasks which you perform sequentially.

- Creating physical volumes from selected storage devices.
- Creating a volume group from physical volumes.
- 3. Configuring logical volumes over the volume group.
- 4. As needed, creating snapshots of logical volumes.

Initializing and Managing Physical Volumes

The following example sets 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
...
```



You might see a message indicating the creation of a system file depending on whether LVM Volume Groups already exist or not.

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 pyremove 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

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

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

The following output is displayed:

```
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'

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 a preferred allocation policy directly when creating a volume group or logical volume.

The vgextend and vgreduce commands add 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.

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
```

The following output is displayed:

```
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.
```

lycreate 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 aren't 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.

Creating and Managing Grouping With Tags

A tag is a string of characters that groups LVM objects of the same type. Only objects in a volume group (VG) can be tagged, such as physical volumes (PV) and logical volumes (LV). PVs lose their tags if they are removed from a VG because tags are stored as part of the VG metadata and are deleted when a PV is removed.

Multiple LVM command can use tags. When you use a tag in a command, the command adds all objects that possess the tag that are of the type expected by its position in the command. You can also use tags to automate actions such as activating all objects within a tagged group.

To create and manage LVM object groupings with tags, do the following:

1. To add a tag to one or more existing objects, use the following command:

```
pvchange --addtag @tag PV
pvchange --addtag @tag VG
pvchange --addtag @tag LV
```

In the previous, *tag* is the name of the tag and should always be prefix with the "@" character. *PV* is one or more physical volume, *VG* is one or more volume group, and *LV* is one or more logical volume. The --addtag option can be repeated to add multiple tags with one command.



Valid tag characters are A-Z a-z 0-9 $_$ + . - / = ! : # & and can be up to 1024 characters. Tags cannot start with a hyphen.

2. To add a tag when creating objects, use the following command:

```
vgcreate --addtag @tag VG
lvcreate --addtag @tag LV
```

3. To delete a tag from one or more existing objects, use the following command:

```
pvchange --deltag @tag PV
pvchange --deltag @tag VG
pvchange --deltag @tag LV
```

The --deltag option can be repeated to add multiple tags with one command.

4. To display tags, use the following command:

```
pvs -o tags PV
pvs -o tags VG
pvs -o tags LV
```

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

Managing Activation and Automatic Activation

You can activate or deactivate logical volumes using the lvchange command. Activating a logical volume makes the logical volume usable through a block device. Deactivating a logical volume makes the logical volume inactive and inaccessible to the kernel.

Additionally, you can enable automatic activation for logical volumes or volume groups, which applies to all logical volumes in the volume group. Automatic activation activates logical volumes in response to attaching an LVM device to a machine. When you attach all physical

volumes in a volume group, the volume group is complete, and the logical volumes in the volume group automatically activate. This automatic behavior can be disabled or enabled using vgchange or lvchange with the --setautoactivation option.

You can also control which logical volumes can be automatically activated by listing which logical volumes can be activated in the /etc/lvm/lvm.conf auto_activation_volume_list parameter.

To manage the activation status of a logical volume, do the following:

1. To deactivate a logical volume, use the following command:

```
lvchange -an VG/LG
```

In the previous, *VG* is the name of the volume group and *LG* is the name of the logical volume to be deactivated.

2. To deactivate all logical volumes in a volume group, use the following command:

```
lvchange -an VG
```

3. To activate a deactivated logical volume, use the following command:

```
lvchange -ay VG/LG
```



 ${\tt systemd} \ automatically \ mounts \ LVM \ volumes \ using \ the \ mount \ points \ specified \ in \\ the \ /etc/fstab \ file.$

4. To activate all deactivated logical volumes in a volume group, use the following command:

```
lvchange -ay VG
```

5. To control which logical volumes can be activated using lvchange commands described above, you can also use the /etc/lvm/lvm.conf configuration file, with the activation/volume_list configuration option. The following example shows that the volume list can specify an entire volume group or just one logical volume within a volume group:

```
volume list = [ "VG", "VG/LV"]
```

You can also use one or more tags to specify which logical devices can be activated. For example:

```
volume list = ["@tag1", "@tag2, "@tag3"]
```

If you leave the brackets empty, then no device can be activated.

```
volume list = []
```

To manage the automatic activation of logical volumes, do the following:



To enable or disable automatic activation for a volume group, use the following command:

```
vgchange --setautoactivation <y|n>
```

2. To enable or disable automatic activation for a logical volume, use the following command:

```
lvchange --setautoactivation <y|n>
```

3. To control automatic activation using the /etc/lvm/lvm.conf configuration file, specify volume groups, or a volume group and a specific logical volume within it, to the activation/auto activation volume list configuration option:

```
auto activation volume list = [ "VG", "VG/LV"]
```

You can also use tags to enable autoactivation. For example:

```
auto_activation_volume_list = ["@tag1", "@tag2, "@tag3"]
```

If you leave the brackets empty, the automatic activation functionality is completely disabled.

```
auto activation volume list = []
```

Creating Logical Volume Snapshots

To create a snapshot of an existing logical volume, use lvcreate --snapshot, 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, 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. Use 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.

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. For example, the following merges the myvg/mylv-snapshot back into myvg/mylv:

```
sudo lvconvert --merge myvg/mylv-snapshot
```

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

```
sudo lvremove myvg/mylv-snapshot
```



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

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.

Configuring and Managing Thinly-Provisioned Logical Volumes

Creating thinly provisioned logical volumes involves two steps:

Create a thin pool.

```
sudo lvcreate --size size --thin vol group/pool name
```

2. Create a thinly provisioned logical volume.

```
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 myvq:

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

Then, the thinly provisioned logical volume mytv is created with a virtual size of 2 GB:

```
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, don't specify the size of the snapshot. Otherwise, its storage would not be provisioned from mytp, for example:

```
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 lycreate(8) and lyresize(8) manual pages.

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
```

config_name

Name of the configuration

fs_type

File system type (ext4 or xfs)

fs name

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.

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.



Working With Software RAID

The Redundant Array of Independent Disks (RAID) feature provides the capability to spread data across several 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 OS. The Oracle Linux kernel uses the multiple device (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.

You can create RAID devices using mdadm or Logical Volume Manager (LVM).

Software 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 doesn't 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 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 (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 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 typically as expensive as or slightly more expensive than RAID-1.

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 doesn't take up as much space as a complete mirror. Write performance is reduced to some extent as a result 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. The double parity is distributed across all the drives in an array, to ensure redundancy at the expense of taking up more space than RAID-5. For example, in an array of four disks, if two disks in the array fails, the parity information is used to reconstruct data. Usable disks are the number of disks minus two. 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.

Creating Software RAID Devices using mdadm

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

```
sudo mdadm --create md device --level=RAID level [options] --raid-devices=Ndevices
```

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, <code>/dev/sd[bcd]</code> 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/sdb, and dev/sdd:

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

The previous example creates a RAID-5 device /dev/md1 out of 3 devices. We can use 4 devices where 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 | sudo tee -a /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/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.

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

```
cat /proc/mdstat
Personalities : [raid1]
mdo : 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.

Creating and Managing Software RAID Devices using LVM

- Ensure that you have created enough physical volumes in a volume group to accommodate the LVM RAID logical volume requirements. For more information about creating physical volumes and volume groups, see Working With Logical Volume Manager.
- 2. Review the raid_fault_policy value in the /etc/lvm/lvm.conf file that specifies how a RAID instance that uses redundant devices reacts to a drive failure. The default value is "warn" which indicates that RAID is configured to log a warning in the system logs. This means that in the event of a device failure, manual action is required to replace the failed device.
- 3. Run the lvcreate command to create the LVM RAID device. See the following sections for examples:
 - RAID Level 0 (Striping) LVM Examples
 - RAID Level 1 (Mirroring) LVM Examples
 - RAID Level 5 (Striping with Distributed Parity) LVM Examples
 - RAID Level 6 (Striping with Double Distributed Parity) LVM Examples
 - RAID Level 10 (Striping of Mirrored Disks) LVM Examples
- Create the filesystem you want on your device. For example, the following command creates an ext4 file system on a RAID 6 logical volume.



5. Consider persisting the logical volume by editing the /etc/fstab file. For example, adding the following line that includes the UUID created in the previous step ensures that the logical volume is remounted after a reboot.

```
UUID=05a78be5-8a8a-44ee-9f3f-1c4764c957e8 /mnt ext4 defaults 0 0
```

For more information about using UUIDs with the /etc/fstab file, see Automatic Device Mappings for Partitions and File Systems .

- 6. If a device failure occurs for LVM RAID levels 5, 6, and 10, ensure that you have a replacement physical volume attached to the volume group that contains the failed RAID device, and do one of the following:
 - Use the following command to switch to a random spare physical volume present in the volume group:

```
sudo lvconvert --repair volume group/logical volume
```

In the previous example, *volume_group* is the volume group and *logical_volume* is the LVM RAID logical volume.

 Use the following command to switch to a specific physical volume present in the volume group:

```
sudo lvconvert --repair volume group/logical volume physical volume
```

In the previous example, *physical_volume* is the specific volume you want to replace the failed physical volume with. For example, /dev/sdb1.

RAID Level 0 (Striping) LVM Examples

Before you can do RAID level 0, you need at least two or more devices. The following example creates a RAID level 0 logical volume mylvraid0 of size 2 GB in the volume group myvg.

```
sudo lvcreate --type raid0 --size 2g --stripes 3 --stripesize 4 -n mylvraid0
myvg
```

The logical volume contains three stripes, which is the number of devices to use in <code>myvg</code> volume group. The stripesize of 4 kilobytes is the size of data that can be written to one device before moving to the next device.

The following output is displayed:

```
Rounding size 2.00~\text{GiB} (512 extents) up to stripe boundary size 2.00~\text{GiB} (513 extents). Logical volume "mylvraid0" created.
```

The lsblk command shows that three out of the four physical volumes are now running the myvg-mylvraid0 RAID 0 logical volume. Also, each instances of myvg-mylvraid0 is a subvolume included in another subvolume containing the data for the RAID logical volume. Each subvolume contains the data subvolume that are labled myvg-mylvraid0_rimage_0, myvg-mylvraid0 rimage 1, and myvg-mylvraid0 rimage 2.

```
lsblk
NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
```



sdb	8:16	0	50G	0 disk
└myvg-mylvraid0_rimage_0	252:2	0	684M	0 lvm
└myvg-mylvraid0	252:5	0	2G	0 lvm
sdc	8:32	0	50G	0 disk
└myvg-mylvraid0_rimage_1	252:3	0	684M	0 lvm
└myvg-mylvraid0	252:5	0	2G	0 lvm
sdd	8:48	0	50G	0 disk
└myvg-mylvraid0_rimage_2	252:4	0	684M	0 lvm
└myvg-mylvraid0	252:5	0	2G	0 lvm
sde	8:64	0	50G	0 disk

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

To remove a RAID 0 logical volume from a volume group, use the lyremove command:

```
sudo lvremove vol group/logical vol
```

Other commands that are available for managing logical volumes include lvchange, lvconvert, lvmdiskscan, lvrename, lvextend, lvreduce, and lvresize.

RAID Level 1 (Mirroring) LVM Examples

Before you can do RAID level 1, you need at least two or more devices. The following example creates a RAID level 1 logical volume mylvraid1 of size 1 GB in the volume group myvg.

```
sudo lvcreate --type raid1 -m 1 --size 1G -n mylvraid1 myvg
```

The following output is displayed:

```
Logical volume "mylvraid1" created.
```

The -m specifies that you want 1 mirror device in the myvg volume group where identical data is written to the first device and second mirror device. You can specify other mirror devices if you want. For example, -m 2 would create two mirrors of the first device. If one device fails the other device mirrors can continue to process requests.

The lsblk command shows that two out of the four available physical volumes are now part of the myvg-mylvraid1 RAID 1 logical volume. Also, each instance of myvg-mylvraid1 includes subvolume pairs for data and metadata. Each data subvolumes are labled myvg-mylvraid1_rimage_0 and myvg-mylvraid1_rimage_1. Each metadata subvolumes are labled myvg-mylvraid1 rmeta 0 and myvg-mylvraid1 rmeta 1.

```
lsblk
NAME
                      MAJ:MIN RM SIZE RO TYPE MOUNTPOIN
                                 50G 0 disk
sdb
                       8:16
 -myvg-mylvraid1 rmeta 0 252:2 0
                                 4M 0 lvm
 └myvg-mylvraid1 252:6 0
                                1G 0 lvm
└myvg-mylvraid1 rimage 0 252:3 0 1G 0 lvm
 \sqsubseteqmyvg-mylvraid1 252:6 0 1G 0 lvm
                      8:32 0 50G 0 disk
-myvg-mylvraid1 rmeta 1 252:4 0 4M 0 lvm
 \sqsubseteqmyvg-mylvraid1 252:6 0 1G 0 lvm
└myvg-mylvraid1 rimage 1 252:5 0 1G 0 lvm
```



└myvg-mylvraid1	252:6	0	1G	0 lvm
sdd	8:48	0	50G	0 disk
sde	8:64	0	50G	0 disk

To display information about logical volumes, use the lvdisplay, lvs, and lvscan commands. For example, you can use the following command to show the synchronization rate between devices in myvg:

To remove a RAID 1 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, lvrename, lvextend, lvreduce, and lvresize.

For example, you can enable the data integrity feature when creating a RAID 1 logical volume using the --raidintegrity y option. This creates subvolumes used to detect and correct data corruption in your RAID images. You can also add or remove this subvolume after creating the logical volume using the following lyconvert command:

```
sudo lvconvert --raidintegrity y myvg/mylvraid1

Creating integrity metadata LV mylvraid1_rimage_0_imeta with size 20.00 MiB.

Logical volume "mylvraid1_rimage_0_imeta" created.

Creating integrity metadata LV mylvraid1_rimage_1_imeta with size 20.00 MiB.

Logical volume "mylvraid1_rimage_1_imeta" created.

Limiting integrity block size to 512 because the LV is active.

Using integrity block size 512 for file system block size 4096.

Logical volume myvg/mylvraid1 has added integrity.
```

You can also use luconvert to split a mirror into individual linear logical volumes. For example, the following command splits the mirror:

```
sudo lvconvert --splitmirror 1 -n lvnewlinear myvg/mylvraid1 Are you sure you want to split raid1 LV myvg/mylvraid1 losing all resilience? [y/n]: y
```

If you had a three instance mirror, the same command would create a two way mirror and a linear logical volume.

You can also add or remove mirrors to an existing mirror. For example, the following command increases a two way mirror to a three way mirror by adding a third mirror:

```
sudo lvconvert -m 2 myvg/mylvraid1
Are you sure you want to convert raid1 LV myvg/mylvraid1 to 3 images
enhancing resilience? [y/n]: y
  Logical volume myvg/mylvraid1 successfully converted.
```



And performing the same command again, but with -m 1 instead deletes the third mirror back down to a two way mirror again and also specify which drive to have removed.

```
sudo lvconvert -m1 myvg/mylvraid1 /dev/sdd
Are you sure you want to convert raid1 LV myvg/mylvraid1 to 2 images reducing
resilience? [y/n]: y
  Logical volume myvg/mylvraid1 successfully converted.
```

For more information, see the lymraid, lycreate, and lyconvert manual pages.

RAID Level 5 (Striping with Distributed Parity) LVM Examples

Before you can do RAID level 5, you need at least three or more devices. The following example creates a RAID level 5 logical volume mylvraid5 of size 1 GB in the volume group myvg.

```
sudo lvcreate --type raid5 -i 2 --size 1G -n mylvraid5 myvg
```

The following output is displayed:

```
Using default stripesize 64.00 KiB. Logical volume "mylvraid5" created.
```

The logical volume contains two stripes, which is the number of devices to use in the <code>myvg</code> volume group. However, the total usable number of devices requires that an extra device be added to account for the parity information. And so, a stripe size of two requires three available drives such that striping and parity information is spread across all three, even though the total usable device space available for striping is only equal to two devices. The parity information across all three devices is enough to deal with the loss of one of the devices.

The stripesize is not specified in the creation command, so the default of 64 kilobytes is used. This is the size of data that can be written to one device before moving to the next device.

The lsblk command shows that three out of the four available physical volumes are now part of the myvg-mylvraid5 RAID 5 logical volume. Also, each instance of myvg-mylvraid5 includes subvolume pairs for data and metadata. Each data subvolumes are labelled myvg-mylvraid5_rimage_0, myvg-mylvraid5_rimage_1, and myvg-mylvraid5_rimage_2. Each metadata subvolumes are labelled myvg-mylvraid5_rmeta_0, myvg-mylvraid5_rmeta_1 and myvg-mylvraid5_rmeta_2.

```
lsblk
NAME
                      MAJ:MIN RM SIZE RO TYPE MOUNTPOIN
                                 50G 0 disk
sdb
                        8:16 0
 -myvg-mylvraid5 rmeta 0 252:2 0
                                 4M 0 lvm
 └myvq-mylvraid5
                     252:8
                             0
                                 1G 0 lvm
└myvg-mylvraid5 rimage 0 252:3 0 512M 0 lvm
 └myvq-mylvraid5
                  252:8 0 1G 0 lvm
                        8:32 0 50G 0 disk
sdc
 -myvg-mylvraid5 rmeta 1 252:4
                             0
                                  4M 0 lvm
 └myvg-mylvraid5 252:8 0
                                 1G 0 lvm
myvg-mylvraid5 rimage 1 252:5 0 512M 0 lvm
 ∟myvq-mylvraid5
                      252:8 0 1G 0 lvm
                        8:48 0
                                 50G 0 disk
├myvg-mylvraid5 rmeta 2 252:6 0 4M 0 lvm
```



To display information about logical volumes, use the <code>lvdisplay</code>, <code>lvs</code>, and <code>lvscan</code> commands. For example, you can use the following command to show the synchronization rate between devices in <code>myvg</code>:

```
sudo lvs -a -o name, copy percent, devices myvg
  T.77
                        Cpy%Sync
Devices
                        25.00
  mylvraid5
mylvraid5 rimage 0(0),mylvraid5_rimage_1(0),mylvraid5_rimage_2(0)
  [mylvraid5 rimage 0]
                                  /dev/
sdf (1)
  [mylvraid5 rimage 1]
                                  /dev/
sdg (1)
  [mylvraid5 rimage 2]
                                  /dev/
sdh (1)
  [mylvraid5 rmeta 0]
                                  /dev/
sdf(0)
  [mylvraid5 rmeta 1]
                                  /dev/
sdg(0)
  [mylvraid5 rmeta 2]
                                  /dev/sdh(0)
```

To remove a RAID 5 logical volume from a volume group, use the lyremove command:

```
sudo lvremove vol group/logical vol
```

Other commands that are available for managing logical volumes include lvchange, lvconvert, lvmdiskscan, lvrename, lvextend, lvreduce, and lvresize.

For example, you can enable the data integrity feature when creating a RAID 5 logical volume using the --raidintegrity y option. This creates subvolumes used to detect and correct data corruption in your RAID images. You can also add or remove this subvolume after creating the logical volume using the following lyconyert command:

```
sudo lvconvert --raidintegrity y myvg/mylvraid5
  Using integrity block size 512 for unknown file system block size, logical
block size 512, physical block size 4096.
  Logical volume myvg/mylvraid5 has added integrity.
```

For more information, see the lymraid, lycreate, and lyconvert manual pages.

RAID Level 6 (Striping with Double Distributed Parity) LVM Examples

Before you can do RAID level 6, you need at least five or more devices. The following example creates a RAID level 6 logical volume mylvraid6 of size 1 GB in the volume group myvg.

```
sudo lvcreate --type raid6 -i 3 -L 1G -n mylvraid6 myvg
```

The following output is displayed:



```
Using default stripesize 64.00 KiB.

Rounding size 1.00 GiB (256 extents) up to stripe boundary size <1.01 GiB (258 extents).

Logical volume "mylvraid6" created.
```

The logical volume contains three stripes, which is the number of devices to use in the myvg volume group. However, the total usable number of devices requires that an extra two devices be added to account for the double parity information. And so, a stripe size of three requires five available drives such that striping and double parity information is spread across all five, even though the total usable device space available for striping is only equal to three devices. The parity information across all five devices is enough to deal with the loss of two of the devices.

The stripesize isn't specified in the creation command, so the default of 64 kilobytes is used. This is the size of data that can be written to one device before moving to the next device.

The lsblk command shows that all five of the available physical volumes are now part of the myvg-mylvraid6 RAID 6 logical volume. Also, each instance of myvg-mylvraid6 includes subvolume pairs for data and metadata. Each data subvolumes are labelled myvg-mylvraid6_rimage_0, myvg-mylvraid6_rimage_1, myvg-mylvraid6_rimage_2, myvg-mylvraid6_rimage_3, and myvg-mylvraid6_rimage_4. Each metadata subvolumes are labelled myvg-mylvraid6_rmeta_0, myvg-mylvraid6_rmeta_1, myvg-mylvraid6_rmeta_2, myvg-mylvraid6 rmeta 3, and myvg-mylvraid6 rmeta 4.

```
lsblk
NAME
                        MAJ:MIN RM SIZE RO TYPE MOUNTPOIN
sdb
                          8:16
                                     50G 0 disk
 -myvg-mylvraid5 rmeta 0 252:2
                                 0
                                      4M 0 lvm
 └mvvg-mvlvraid5
                        252:12
                                     1G 0 lvm
└myvg-mylvraid5 rimage 0 252:3
                                   344M 0 lvm
                                 0
  └myvg-mylvraid5
                        252:12
                                 0
                                     1G 0 lvm
                                     50G 0 disk
                          8:32
                                 0
sdc
 -myvg-mylvraid5 rmeta 1 252:4
                                 0
                                      4M 0 lvm
  └myvg-mylvraid5
                        252:12
                                 0
                                     1G 0 lvm
 -myvg-mylvraid5 rimage 1 252:5
                                 0 344M 0 lvm
  └myvg-mylvraid5
                        252:12
                                 0
                                     1G 0 lvm
                          8:48
                                     50G 0 disk
                                 0
sdd
 -myvg-mylvraid5 rmeta 2 252:6
                                 0
                                     4M 0 lvm
  └myvg-mylvraid5
                        252:12
                                Ω
                                     1G 0 lvm
└myvg-mylvraid5 rimage 2 252:7
                                 0 344M 0 lvm
  └myvg-mylvraid5
                        252:12
                                0
                                     1G 0 lvm
                          8:64
sde
                                 0
                                     50G 0 disk
-myvg-mylvraid5 rmeta 3 252:8
                                 0
                                     4M 0 lvm
  └myvg-mylvraid5
                        252:12
                                 0
                                     1G 0 lvm
 -myvg-mylvraid5 rimage 3 252:9
                                   344M 0 lvm
                                0
  └myvg-mylvraid5
                        252:12
                               0
                                     1G 0 lvm
sdf
                          8:80 0
                                     50G 0 disk
 -myvg-mylvraid5 rmeta 4 252:10
                                      4M 0 lvm
                               0
  └myvg-mylvraid5
                        252:12
                                 0
                                     1G 0 lvm
 -myvg-mylvraid5 rimage 4 252:11
                               0 344M 0 lvm
  —myvg-mylvraid5 252:12
                                     1G 0 lvm
```



To display information about logical volumes, use the <code>lvdisplay</code>, <code>lvs</code>, and <code>lvscan</code> commands. For example, you can use the following command to show the synchronization rate between devices in <code>myvg</code>:

```
sudo lvs -a -o name, sync percent, devices myvg
  T.\7
                        Cpy%Sync
Devices
  mylvraid6
                        31.26
mylvraid6 rimage 0(0), mylvraid6 rimage 1(0), mylvraid6 rimage 2(0), mylvraid6 ri
mage 3(0), mylvraid6 rimage 4(0)
  [mylvraid6 rimage 0]
                                  /dev/
sdf (1)
  [mylvraid6 rimage 1]
                                  /dev/
sdq (1)
  [mylvraid6 rimage 2]
                                  /dev/
sdh(1)
  [mylvraid6 rimage 3]
                                  /dev/
sdi(1)
  [mylvraid6 rimage 4]
                                  /dev/
sdj (1)
  [mylvraid6 rmeta 0]
                                  /dev/
sdf(0)
  [mylvraid6 rmeta 1]
                                  /dev/
sdq(0)
  [mylvraid6 rmeta 2]
                                  /dev/
sdh(0)
  [mylvraid6 rmeta 3]
                                  /dev/
sdi(0)
  [mylvraid6 rmeta 4]
                                  /dev/sdj(0)
```

To remove a RAID 6 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, lvrename, lvextend, lvreduce, and lvresize.

For example, you can enable the data integrity feature when creating a RAID 6 logical volume using the --raidintegrity y option. This creates subvolumes used to detect and correct data corruption in the RAID images. You can also add or remove this subvolume after creating the logical volume using the following lyconyert command:

```
sudo lvconvert --raidintegrity y myvg/mylvraid6
Using integrity block size 512 for file system block size 4096.
Logical volume myvg/mylvraid6 has added integrity.
```

For more information, see the lymraid, lycreate, and lyconvert manual pages.

RAID Level 10 (Striping of Mirrored Disks) LVM Examples

Before you can do RAID level 10, you need at least four or more devices. The following example creates a RAID level 10 logical volume mylvraid10 of size 10 GB in the volume group myvg.

```
sudo lvcreate --type raid10 -i 2 -m 1 --size 10G -n mylvraid10 myvg
```

The following output is displayed:

```
Logical volume "mylvraid10" created.
```

The -m specifies that you want 1 mirror device in the myvg volume group where identical data is written to pairs of mirrored device sets which are also using striping across the sets. Logical volume data remains available if one or more devices remains in each mirrored device set.

The lsblk command shows that four out of the five available physical volumes are now part of the myvg-mylvraid10 RAID 10 logical volume. Also, each instance of myvg-mylvraid10 includes subvolume pairs for data and metadata. Each data subvolumes are labelled myvg-mylvraid10_rimage_0, myvg-mylvraid10_rimage_1, myvg-mylvraid10_rimage_2, and myvg-mylvraid10_rimage_3. Each metadata subvolumes are labelled myvg-mylvraid10_rmeta_0, myvg-mylvraid10_rmeta_1, myvg-mylvraid10_rmeta_2, and myvg-mylvraid10_rmeta_3.

lsblk						
NAME	MAJ:MIN	RM	SIZE	RO	TYPE	MOUNTPOIN
sdb	8:16	0	50G	0	disk	
-myvg-mylvraid10_rmeta_0	252:2	0	4 M	0 1	lvm	
│ └myvg-mylvraid10	252:10	0	10G	0	lvm	
└myvg-mylvraid10_rimage_(252:3	0	5G	0	lvm	
└myvg-mylvraid10	252:10	0	10G	0	lvm	
sdc	8:32	0	50G	0	disk	
-myvg-mylvraid10_rmeta_1	252:4	0	4 M	0 1	lvm	
└myvg-mylvraid10	252:10	0	10G	0	lvm	
└myvg-mylvraid10_rimage_1	1 252:5	0	5G	0	lvm	
└myvg-mylvraid10	252:10	0	10G	0	lvm	
sdd	8:48	0	50G	0	disk	
-myvg-mylvraid10_rmeta_2	252:6	0	4 M	0 1	lvm	
└myvg-mylvraid10	252:10	0	10G	0	lvm	
└myvg-mylvraid10_rimage_2	2 252:7	0	5G	0	lvm	
└myvg-mylvraid10	252:10	0	10G	0	lvm	
sde	8:64	0	50G	0	disk	
-myvg-mylvraid10_rmeta_3	252:8	0	4 M	0 1	lvm	
└myvg-mylvraid10	252:10	0	10G	0	lvm	
—myvg-mylvraid10_rimage_3	3 252:9	0	5G	0	lvm	
└myvg-mylvraid10	252:10	0	10G	0	lvm	
sdf	8:80	0	50G	0	disk	



To display information about logical volumes, use the <code>lvdisplay</code>, <code>lvs</code>, and <code>lvscan</code> commands. For example, you can use the following command to show the synchronization rate between devices in <code>myvg</code>:

```
sudo lvs -a -o name, sync percent, devices myvg
                          Cpy%Sync
Devices
  mylvraid101
                          68.82
mylvraid10 rimage 0(0), mylvraid10 rimage 1(0), mylvraid10 rimage 2(0), mylvraid1
0 rimage 3(0)
  [mylvraid10 rimage 0]
                                   /dev/
sdf (1)
  [mylvraid10 rimage 1]
                                   /dev/
sdq (1)
  [mylvraid10 rimage 2]
                                   /dev/
sdh(1)
  [mylvraid10 rimage 3]
                                   /dev/
sdi(1)
  [mylvraid10 rmeta 0]
                                   /dev/
sdf(0)
  [mylvraid10 rmeta 1]
                                   /dev/
sdq(0)
  [mylvraid10 rmeta 2]
                                   /dev/
sdh(0)
  [mylvraid10 rmeta 3]
                                   /dev/sdi(0)
```

To remove a RAID 10 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, lvrename, lvextend, lvreduce, and lvresize.

For example, you can enable the data integrity feature when creating a RAID 10 logical volume using the --raidintegrity y option. This creates subvolumes used to detect and correct data corruption in your RAID images. You can also add or remove this subvolume after creating the logical volume using the following lvconvert command:

```
sudo lvconvert --raidintegrity y myvg/mylvraid10
  Using integrity block size 512 for unknown file system block size, logical
block size 512, physical block size 4096.
  Logical volume myvg/mylvraid10 has added integrity.
```

For more information, see the lymraid, lycreate, and lyconvert manual pages.



Using Encrypted Block Devices

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

About Encrypted Block Devices

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

Creating Encrypted Volumes

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

LUKS is an encryption specification that implements a platform independent and standard ondisk format. The standard ensures interoperability and compatibility among different distributions and programs. The implementation also includes tools that would simplify the administration of the encrypted disks. If used, this feature requires a passphrase at boot time. The correct passphrase then unlocks the encryption key to enable volume decryption.

For more information about LUKS, see the https://gitlab.com/cryptsetup/cryptsetup/blob/master/README.md file.

To encrypt volumes with LUKS, follow these steps:

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

```
sudo cryptsetup luksFormat /dev/sdd
```

The following warning is displayed:

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

Open the device and create the device mapping, for example:

```
sudo cryptsetup luksOpen /dev/sdd cryptfs
```

You're prompted to enter the passphrase:

```
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 OS 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
```

The following output is displayed:

```
/dev/mapper/cryptfs is active.
type: LUKS2
cipher: aes-xts-plain64
keysize: 512 bits
key location: keyring
device: /dev/sdd
sector size: 4096
offset: 32768 sectors
size: 104824832 sectors
mode: read/write
```

To remove the device mapping:

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

For example, for the encrypted volume <code>cryptfs</code>, use the following command:

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

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



7

Working With Linux I-O Storage

Oracle Linux 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. Note that Mellanox InfiniBand is only supported with UEK. You can install the targetcli package by running:

```
sudo dnf install -y targetcli
```

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. You can install the fcoe-utils package by running:

sudo dnf install -y fcoe-utils

About iSCSI Devices

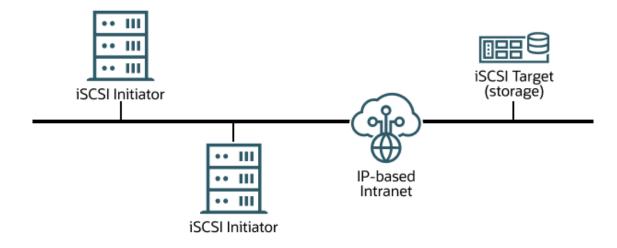
The Internet Small Computer System Interface (iSCSI) is an IP-based standard for connecting storage devices. iSCSI encapsulates SCSI commands in IP network packets to enable data transfer over long distances and sharing of storage by client systems. iSCSI uses the existing IP infrastructure and doesn't require the 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 7-1 shows a network where several iSCSI initiators can access the shared storage that's attached to an iSCSI target.

Figure 7-1 iSCSI Initiators and an iSCSI Target Connected over an IP-based Network



A hardware-based iSCSI initiator uses a dedicated iSCSI HBA. Oracle Linux enables 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 isn't available at the level of the system BIOS. Thus, you can't 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 doesn't support TOE, although suitable drivers might be available directly from some card vendors.

For more information about LIO, see https://docs.kernel.org/target/tcmu-design.html.

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:

```
sudo targetcli
targetcli shell version 2.1.58
Copyright 2011-2013 by Datera, Inc and others.
For help on commands, type 'help'.
```

2. (Optional) Use the ls command to 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.



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.
```

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

6. 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.
```

7. 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.
```

The default TCP port number is 3260.

Note:

/iscsi> ls

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:

```
/iscsi/iqn.20.../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:
 - Add the port:

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

Add the target:

```
sudo firewall-cmd --permanent --add-service \
iqn.2013-01.com.mydom.host01.x8664:sn.ef8e14f87344
```

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

```
/iscsi/iqn.20.../tpg1/portals> ls /
```

```
| 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- 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]
```

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:



Caution:

The demonstration mode is inherently insecure. For information about configuring secure authentication modes, see the targetcli(8) manual page.

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

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 2.1.58
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

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:

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

The following output is displayed:

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

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



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

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

```
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 -1

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
```

The following output is displayed:

```
iSCSI Transport Class version 2.0-870
version 6.2.1.9
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 -1 | 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 <code>/dev/disk/by-path</code>, which is displayed by using the following command:

```
ls -1 /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 $\protect\operatorname{var/log/messages}$ file, for example:

```
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 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 mounting the file system until the network has been enabled.



When adding iSCSI LUN entries to /etc/fstab, see 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 if the target continues to serve those LUNs and you don't 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, use the iscsiadm command on an iSCSI initiator to update the entries in its discovery database. The following example assumes that the target uses the SendTargets discovery method

To add new records that aren't 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.

8

Working With NVMe Storage Over Fibre Channel

Oracle Linux uses the Non-volatile Memory Express over Fibre Channel (NVMe/FC) protocol to manage hosts and NVMe solid state storage devices over a fibre channel network. Oracle Linux running on machines that have NVMe/FC enabled host bus adapters (HBAs) can discover and connect to NVMe SSD devices attached to a fibre network. You can establish connectivity from hosts to these SSD devices through these adapters by using various nvme commands which are part of the nvme-cli package. nvme-cli is an NVMe storage CLI utility available on Oracle Linux.

This chapter provides instructions relating to configuring HBA devices such as certain Broadcom Emulex or Marvell Qlogic Fibre Channel adapters to connect to remote NVMe SSD devices over a Fibre channel network.

About NVMe/FC Devices

The NVMe/FC is a protocol for managing NVMe hosts and NVMe SSD storage devices on a Fibre channel network. NVMe/FC requires infrastructure such as fiber-optic cabling, NVMe/FC enabled interface adapters, and NVMe/FC enabled network switches. Fibre channel networks typically provide better throughput, lower latency, and higher performance when compared to iSCSI.

A host (often called the NVMe initiator) accesses the storage targets over a fibre channel network. The host can be a server, a VM, or any device equipped with an NVMe/FC adapter or software stack that can initiate requests to access remote NVMe/FC storage. To an NVMe initiator, the storage appears to be locally attached.

NVMe initiators and targets use dedicated NVMe HBAs which are NVMe adapter with ports through which it connects to a fibre channel network.

The parameters in the nvme connect command that represent the initiator and the target include the following:

- host-traddr: This parameter specifies the host transport address and refers to the address
 of the initiator. The address or identifier is the host's fibre channel HBA port through which
 it connects to the fibre channel fabric. This port is typically called the local port.
- traddr: This parameter specifies the transport address and refers to the address of the target server or controller hosting one or more NVMe SSD storage device. This port is typically called the remote port.

Local and remote port addresses are a combination of the following elements:

- World Wide Namespace Name (WWNN): This is a globally unique identifier assigned to each namespace within an NVMe subsystem. The NVMe protocol uses WWNNs when routing commands to correct namespaces.
- World Wide Namespace Port (WWNP): This is a globally unique identifier that specifies the
 port through which a namespace is accessed. The NVMe protocol uses WWNP to route
 commands to the appropriate namespace device.

You can query local and remote port address combinations to discover available target namespaces that represent the NVMe storage devices. After identifying the namepace of a storage device, you can then combine the local and remote port addresses with the device's namespace to connect the host to the target NVMe storage device.

For more information about the NVMe specification, see https://nvmexpress.org/specifications/.

NVMe/FC Storage Adapter Compatibility

The NVMe/FC discovery and disconnect feature is compatible with the Fibre Channel storage adapters listed in the following document: https://linux.oracle.com/ Component_Compatibility_Guide.pdf.

Install NVMe-CLI and Identify Host Namespace

The following procedure describes how to install the nvme-cli NVMe CLI utility and identify the host namespace.

On the host open the nvme-cli command line utility:

```
sudo dnf install nvme-cli
```

2. Verify that a namespace exists for the host:

```
nvme show-hostngn
```



Alternatly, you can find the namespace in the /etc/nvme/hostnqn file created when you installed nvme-cli.

Discovering and Connecting to Targets from Broadcom Hosts

The following procedure describes how to discover and connect to an NVMe/FC target from an Oracle Linux host using a Broadcom Emulex Fibre Channel adapter.

1. Identify the WWNN and WWPN of the local and remote ports in the /sys/class/ scsi_host/host*/nvme_info directory. For example, the following shows a host with one local port and two remote ports:

- 2. Apply the following format change to a combination of the WWPN and WWNN from the local port and one of the remote ports.
 - For all WWPNs, append np-0 to the beginning of the address.
 - For all WWNNs, append nn-0 to the beginning of the address.

For example,

```
LPORT WWNN nn-0x200000109acfgcad WWPN pn-0x100000109acfgcad RPORT WWNN nn-0x2014c146dc13a14b WWPN pn-0x2015c146dc13a14b
```

3. Run a discovery command to identify NVMe SSD devices available on the remote port. Use the following command format:

```
nvme discover --transport fc --traddr <rportwwnn>:<rportwwnp> --host-
traddr
```

In the previous, <rportwwnn> and <rportwwnp> are the remote port world wide namespace
name and port, and <rportwwnp> are the local world wide namespace
name and port.

For example, the following command

```
# nvme discover --transport fc --traddr
nn-0x2014c146dc13a14b:pn-0x2015c146dc13a14b --host-traddr
nn-0x200000109acfgcad:pn-0x100000109acfgcad

Discovery Log Number of Records 1, Generation counter 2
====Discovery Log Entry 0======
trtype: fc
adrfam: fibre-channel
subtype: nvme subsystem
treq: not specified
portid: 0
trsvcid: none
subnqn:
nqn.1432-01.com.netapp:sn.c44dd3bsce4245edc1adc021cb11a0f6:test_broadcom
traddr: nn-0x2014c146dc13a14b:pn-0x2015c146dc13a14b
```



4. Establish a connection to the device represented by the subnqn listed in the discovery response. Use the following command format:

```
nvme connect --transport fc --traddr <rportwwnn>:<rportwwnp> --host-traddr
<lportwwnn>:<lportwwnp> -n <subnqn> -k <s>
```

In the previous,

- <rportwwnn> and <rportwwnp> are the remote port world wide namespace name and port,
- <rportwwnn> and <rportwwnp> are the local world wide namespace name and port,
- <subnqn> is the value of the subnqn parameter generated by the discovery command,
- <s> is the keep-alive time in seconds for the command to wait for a response from the remote NVMe device before the command generates a timeout message.

For example, the following command establishes a connection to a remote port with a 5 second timer.

```
nvme connect --transport fc --traddr
nn-0x2014c146dc13a14b:pn-0x2015c146dc13a14b --host-traddr
nn-0x200000109acfgcad:pn-0x100000109acfgcad -n
nqn.1432-01.com.netapp:sn.c44dd3bsce4245edc1adc021cb11a0f6:test_broadcom -k 5
```

5. Verify that the storage is now available for the host to use. For example:

```
nvme list -v
Subsystem
          Subsystem-
NQN
       Controllers
_____
nvme-subsvs1
ngn.1432-01.com.netapp:sn.c44dd3bsce4245edc1adc021cb11a0f6:test broadcom
                 nvme1
Device SN
                   MN
    TxPort Asdress
                   Slot Subsystem Namespaces
______
------ ----- ------ ------
nvme1 81EYIJSh2VMMAAAAAAB NetApp ONTAP Controller
FFFFFFFF fc
          traddr nn-0x2014c146dc13a14b:pn-0x2015c146dc13a14b,host-
traddr nn-0x200000109acfgcad:pn-0x100000109acfgcad nvme-subsys1 nvme1n1
Device Generic NSID
                        Usage
          Controllers
Format.
______
/dev/nvmeln1 /dev/ngln1 0x1 107.37 GB / 107.37 GB 4 KiB +
0 B nvme1
```



Discovering and Connecting to Targets from Qlogic Hosts

The following procedure describes how to discover and connect to an NVMe/FC target from an Oracle Linux host using a Marvell Qlogic Fibre Channel adapter.

1. Ensure that the qla2xxx Qlogic module is installed and loaded.

```
modprobe -r qla2xxx
modprobe qla2xxx
```

2. Identify the WWNN and WWPN of the local and remote ports. Use the following command:

```
sudo dmesg | grep traddr
```

For example, the following shows a host with one local port and one remote ports:

```
sudo dmesg | grep traddr

[ 6.139862] qla2xxx [0000:04:00.0]-fffff:0: register_localport: host-
traddr=nn-0x200000109acfgcad:pn-0x100000109acfgcad on portID:10700
[ 6.241762] qla2xxx [0000:04:00.0]-2102:0: qla_nvme_register_remote:
traddr=nn-0x2014c146dc13a14b:pn-0x2015c146dc13a14b PortID:01050d
```

3. Run a discovery command to identify NVMe SSD devices available on the remote port. Use the following command format:

```
nvme discover --transport fc --traddr <rportwwnn>:<rportwwnp> --host-
traddr <lportwwnn>:<lportwwnp>
```

In the previous, <rportwwnn> and <rportwwnp> are the remote port world wide namespace
name and port, and <rportwwnp> are the local world wide namespace
name and port.

For example, the following command

```
# nvme discover --transport fc --traddr
nn-0x2014c146dc13a14b:pn-0x2015c146dc13a14b --host-traddr
nn-0x200000109acfgcad:pn-0x100000109acfgcad

Discovery Log Number of Records 1, Generation counter 2
====Discovery Log Entry 0=====
trtype: fc
adrfam: fibre-channel
subtype: nvme subsystem
treq: not specified
portid: 0
trsvcid: none
subnqn:
nqn.1432-01.com.netapp:sn.c44dd3bsce4245edc1adc021cb11a0f6:test_qlogic
traddr: nn-0x2014c146dc13a14b:pn-0x2015c146dc13a14b
```



4. Establish a connection to the device represented by the subnqn listed in the discovery response. Use the following command format:

```
nvme connect --transport fc --traddr <rportwwnn>:<rportwwnp> --host-traddr
<lportwwnn>:<lportwwnp> -n <subnqn> -k <s>
```

In the previous,

- rportwwnn> and are the remote port world wide namespace name and port,
- and

 are the local world wide namespace name and port,
- <subnqn> is the value of the subnqn parameter generated by the discovery command,
- <s> is the keep-alive time in seconds for the command to wait for a response from the remote NVMe device before the command generates a timeout message.

For example, the following command establishes a connection to a remote port with a 5 second timer.

```
nvme connect --transport fc --traddr
nn-0x2014c146dc13a14b:pn-0x2015c146dc13a14b --host-traddr
nn-0x200000109acfgcad:pn-0x100000109acfgcad -n
nn-0x2014c146dc13a14b:pn-0x2015c146dc13a14b:test qlogic -k 5
```

5. Verify that the storage is now available for the host to use. For example:

```
nvme list -v
Subsystem
           Subsystem-
NON
        Controllers
nvme-subsvs1
ngn.1432-01.com.netapp:sn.c44dd3bsce4245edc1adc021cb11a0f6:test glogic
                nvme1
Device SN
                      MN
     TxPort Asdress
                     Slot Subsystem Namespaces
______
______ _______
     81EYIJSh2VMMAAAAAAAB NetApp ONTAP Controller
FFFFFFFF fc traddr nn-0x2014c146dc13a14b:pn-0x2015c146dc13a14b,host-
traddr nn-0x200000109acfgcad:pn-0x100000109acfgcad nvme-subsys1 nvme1n1
        Generic
                 NSID
Device
                        Usage
         Controllers
/dev/nvmeln1 /dev/ngln1 0x1 107.37 GB / 107.37 GB 4 KiB +
0 B nvme1
```



9

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

Device Multipathing Sample Setup

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 9-1 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 hbal on the server and controller ctrll on the storage array.
- Between host bus adapter hba2 on the server and controller ctrl2 on the storage array.

Mapped by DM-Multipa Application Server /dev/mapper/mpath1 hba1 hba2 multipathd SAN /dev/sdc /dev/sd: ctr12 ctr11 Disk appears Disk to the server as /dev/sdc or Storage /dev/sdf depending on the path

Figure 9-1 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.

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.



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) in /dev/mapper/WWID. WWIDs are unique to their respective devices.

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 Working With the Multipathing Configuration File.

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

```
sudo mpathconf
```

Information similar to the following is displayed:

```
multipath is enabled
find_multipaths is enabled
user_friendly_names is enabled
dm_multipath modules is loaded
multipathd is running
```

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

You can use the multipath device in <code>/dev/mapper</code> 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.

Configuring Multipathing

1. Install the device-mapper-multipath package.

```
sudo dnf install device-mapper-multipath
```

2. Activate 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 Working With the Multipathing Configuration File.

To display the current multipath configuration, run multipath -11:

```
sudo multipath -11
```

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 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), multipath(8), multipath.conf(5), and scsi id(8) manual pages.

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. 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. 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. Excluded devices cannot be subsumed by a multipath device.

The example shows different ways that you can use to exclude devices: by WWID (wwid) and by device name (devnode).

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's 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 <code>vendor</code>, <code>product</code>, and optional <code>revision</code> settings, which must match the information in <code>sysfs</code> for the device.

To add a storage device that DM-Multipath doesn't list as being supported, obtain the vendor, product, and revision information from the <code>vendor</code>, <code>model</code>, and <code>rev</code> files under <code>/sys/block/device</code> <code>name/device</code>.

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

```
defaults {
    user_friendly_names yes
```



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



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.

