



Red Hat Enterprise Linux 9

Managing storage devices

Deploying and configuring single-node storage in Red Hat Enterprise Linux 9

Red Hat Enterprise Linux 9 Managing storage devices

Deploying and configuring single-node storage in Red Hat Enterprise Linux 9

Legal Notice

Copyright © 2022 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux[®] is the registered trademark of Linus Torvalds in the United States and other countries.

Java[®] is a registered trademark of Oracle and/or its affiliates.

XFS[®] is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL[®] is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js[®] is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack[®] Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

Abstract

This documentation provides instructions on how to effectively manage storage devices in Red Hat Enterprise Linux 9.

Table of Contents

MAKING OPEN SOURCE MORE INCLUSIVE	4
PROVIDING FEEDBACK ON RED HAT DOCUMENTATION	5
CHAPTER 1. AVAILABLE STORAGE OPTIONS OVERVIEW	6
1.1. LOCAL STORAGE OVERVIEW	6
1.2. REMOTE STORAGE OVERVIEW	8
1.3. GFS2 FILE SYSTEM OVERVIEW	8
1.4. GLUSTER STORAGE OVERVIEW	9
1.5. CEPH STORAGE OVERVIEW	9
CHAPTER 2. DISK PARTITIONS	11
2.1. OVERVIEW OF PARTITIONS	11
2.2. CONSIDERATIONS BEFORE MODIFYING PARTITIONS ON A DISK	11
2.3. COMPARISON OF PARTITION TABLE TYPES	12
2.4. MBR DISK PARTITIONS	12
2.5. EXTENDED MBR PARTITIONS	14
2.6. MBR PARTITION TYPES	14
2.7. GUID PARTITION TABLE	16
2.8. PARTITION TYPES	17
2.9. PARTITION NAMING SCHEME	18
2.10. MOUNT POINTS AND DISK PARTITIONS	19
CHAPTER 3. GETTING STARTED WITH PARTITIONS	20
3.1. CREATING A PARTITION TABLE ON A DISK WITH PARTED	20
3.2. VIEWING THE PARTITION TABLE WITH PARTED	21
3.3. CREATING A PARTITION WITH PARTED	22
3.4. SETTING A PARTITION TYPE WITH FDISK	23
3.5. RESIZING A PARTITION WITH PARTED	24
3.6. REMOVING A PARTITION WITH PARTED	26
CHAPTER 4. STRATEGIES FOR REPARTITIONING A DISK	28
4.1. USING UNPARTITIONED FREE SPACE	28
4.2. USING SPACE FROM AN UNUSED PARTITION	28
4.3. USING FREE SPACE FROM AN ACTIVE PARTITION	29
4.3.1. Destructive repartitioning	29
4.3.2. Non-destructive repartitioning	30
CHAPTER 5. CONFIGURING AN ISCSI TARGET	33
5.1. INSTALLING TARGETCLI	33
5.2. CREATING AN ISCSI TARGET	34
5.3. ISCSI BACKSTORE	35
5.4. CREATING A FILEIO STORAGE OBJECT	35
5.5. CREATING A BLOCK STORAGE OBJECT	36
5.6. CREATING A PSCSI STORAGE OBJECT	37
5.7. CREATING A MEMORY COPY RAM DISK STORAGE OBJECT	37
5.8. CREATING AN ISCSI PORTAL	38
5.9. CREATING AN ISCSI LUN	39
5.10. CREATING A READ-ONLY ISCSI LUN	40
5.11. CREATING AN ISCSI ACL	41
5.12. SETTING UP THE CHALLENGE-HANDSHAKE AUTHENTICATION PROTOCOL FOR THE TARGET	43
5.13. REMOVING AN ISCSI OBJECT USING TARGETCLI TOOL	43

CHAPTER 6. CONFIGURING AN ISCSI INITIATOR	45
6.1. CREATING AN ISCSI INITIATOR	45
6.2. SETTING UP THE CHALLENGE-HANDSHAKE AUTHENTICATION PROTOCOL FOR THE INITIATOR	46
6.3. MONITORING AN ISCSI SESSION USING THE ISCSIADM UTILITY	47
6.4. DM MULTIPATH OVERRIDES OF THE DEVICE TIMEOUT	48
CHAPTER 7. USING FIBRE CHANNEL DEVICES	49
7.1. RESIZING FIBRE CHANNEL LOGICAL UNITS	49
7.2. DETERMINING THE LINK LOSS BEHAVIOR OF DEVICE USING FIBRE CHANNEL	49
7.3. FIBRE CHANNEL CONFIGURATION FILES	50
7.4. DM MULTIPATH OVERRIDES OF THE DEVICE TIMEOUT	51
CHAPTER 8. NVME OVER FABRICS USING RDMA	52
8.1. OVERVIEW OF NVME OVER FABRIC DEVICES	52
8.2. SETTING UP AN NVME/RDMA TARGET USING CONFIGFS	52
8.3. SETTING UP THE NVME/RDMA TARGET USING NVMETCLI	54
8.4. CONFIGURING AN NVME/RDMA CLIENT	55
CHAPTER 9. NVME OVER FABRICS USING FC	57
9.1. OVERVIEW OF NVME OVER FABRIC DEVICES	57
9.2. CONFIGURING THE NVME INITIATOR FOR BROADCOM ADAPTERS	57
9.3. CONFIGURING THE NVME INITIATOR FOR QLOGIC ADAPTERS	59
CHAPTER 10. ENABLING MULTIPATHING ON NVME DEVICES	61
10.1. NATIVE NVME MULTIPATHING AND DM MULTIPATH	61
10.2. ENABLING NATIVE NVME MULTIPATHING	61
10.3. ENABLING DM MULTIPATH ON NVME DEVICES	63
CHAPTER 11. GETTING STARTED WITH SWAP	67
11.1. OVERVIEW OF SWAP SPACE	67
11.2. RECOMMENDED SYSTEM SWAP SPACE	67
11.3. EXTENDING SWAP ON AN LVM2 LOGICAL VOLUME	68
11.4. CREATING AN LVM2 LOGICAL VOLUME FOR SWAP	69
11.5. CREATING A SWAP FILE	70
11.6. REDUCING SWAP ON AN LVM2 LOGICAL VOLUME	71
11.7. REMOVING AN LVM2 LOGICAL VOLUME FOR SWAP	71
11.8. REMOVING A SWAP FILE	72
CHAPTER 12. CONFIGURING FIBRE CHANNEL OVER ETHERNET	73
12.1. USING HARDWARE FCOE HBAS IN RHEL	73
12.2. SETTING UP A SOFTWARE FCOE DEVICE	73
CHAPTER 13. MANAGING TAPE DEVICES	76
13.1. TYPES OF TAPE DEVICES	76
13.2. INSTALLING TAPE DRIVE MANAGEMENT TOOL	76
13.3. WRITING TO REWINDING TAPE DEVICES	76
13.4. WRITING TO NON-REWINDING TAPE DEVICES	78
13.5. SWITCHING TAPE HEAD IN TAPE DEVICES	79
13.6. RESTORING DATA FROM TAPE DEVICES	80
13.7. ERASING DATA FROM TAPE DEVICES	80
13.8. TAPE COMMANDS	81

MAKING OPEN SOURCE MORE INCLUSIVE

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

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

We appreciate your feedback on our documentation. Let us know how we can improve it.

Submitting comments on specific passages

1. View the documentation in the **Multi-page HTML** format and ensure that you see the **Feedback** button in the upper right corner after the page fully loads.
2. Use your cursor to highlight the part of the text that you want to comment on.
3. Click the **Add Feedback** button that appears near the highlighted text.
4. Add your feedback and click **Submit**.

Submitting feedback through Bugzilla (account required)

1. Log in to the [Bugzilla](#) website.
2. Select the correct version from the **Version** menu.
3. Enter a descriptive title in the **Summary** field.
4. Enter your suggestion for improvement in the **Description** field. Include links to the relevant parts of the documentation.
5. Click **Submit Bug**.

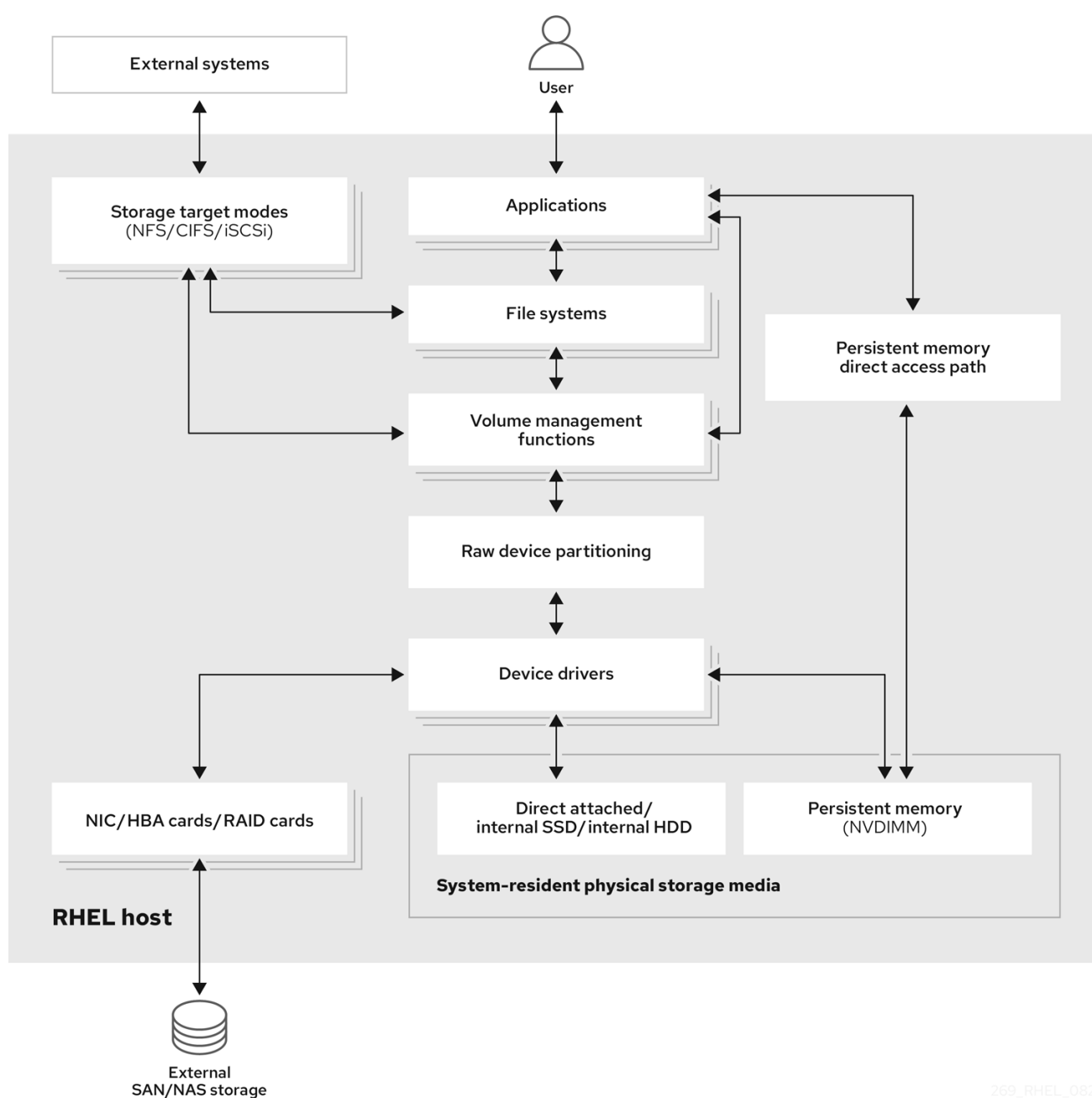
CHAPTER 1. AVAILABLE STORAGE OPTIONS OVERVIEW

There are several local, remote, and cluster-based storage options available on Red Hat Enterprise Linux 8.

Local storage implies that the storage devices are either installed on the system or directly attached to the system.

With remote storage, devices are accessed over LAN, the internet, or using a Fibre channel network. The following high level Red Hat Enterprise Linux storage diagram describes the different storage options.

Figure 1.1. High level Red Hat Enterprise Linux storage diagram



269_RHEL_0822

1.1. LOCAL STORAGE OVERVIEW

Red Hat Enterprise Linux 9 offers several local storage options.

Basic disk administration

Using **parted** and **fdisk**, you can create, modify, delete, and view disk partitions. The following are the partitioning layout standards:

Master Boot Record (MBR)

It is used with BIOS-based computers. You can create primary, extended, and logical partitions.

GUID Partition Table (GPT)

It uses Globally Unique identifier (GUID) and provides unique disk and partition GUID.

To encrypt the partition, you can use Linux Unified Key Setup-on-disk-format (LUKS). To encrypt the partition, select the option during the installation and the prompt displays to enter the passphrase. This passphrase unlocks the encryption key.

Storage consumption options

Non-Volatile Dual In-line Memory Modules (NVDIMM) Management

It is a combination of memory and storage. You can enable and manage various types of storage on NVDIMM devices connected to your system.

Block Storage Management

Data is stored in the form of blocks where each block has a unique identifier.

File Storage

Data is stored at file level on the local system. These data can be accessed locally using XFS (default) or ext4, and over a network by using NFS and SMB.

Logical volumes

Logical Volume Manager (LVM)

It creates logical devices from physical devices. Logical volume (LV) is a combination of the physical volumes (PV) and volume groups (VG). Configuring LVM include:

- Creating PV from the hard drives.
- Creating VG from the PV.
- Creating LV from the VG assigning mount points to the LV.

Virtual Data Optimizer (VDO)

It is used for data reduction by using deduplication, compression, and thin provisioning. Using LV below VDO helps in:

- Extending of VDO volume
- Spanning VDO volume over multiple devices

Local file systems

XFS

The default RHEL file system.

Ext4

A legacy file system.

Stratis

It is available as a Technology Preview. Stratis is a hybrid user-and-kernel local storage management system that supports advanced storage features.

1.2. REMOTE STORAGE OVERVIEW

Following are the remote storage options available in Red Hat Enterprise Linux 8:

Storage connectivity options

iSCSI

RHEL 9 uses the `targetcli` tool to add, remove, view, and monitor iSCSI storage interconnects.

Fibre Channel (FC)

RHEL 9 provides the following native Fibre Channel drivers:

- **lpfc**
- **qla2xxx**
- **Zfcp**

Non-volatile Memory Express (NVMe)

An interface which allows host software utility to communicate with solid state drives. Use the following types of fabric transport to configure NVMe over fabrics:

- NVMe over fabrics using Remote Direct Memory Access (RDMA).
- NVMe over fabrics using Fibre Channel (FC)

Device Mapper multipathing (DM Multipath)

Allows you to configure multiple I/O paths between server nodes and storage arrays into a single device. These I/O paths are physical SAN connections that can include separate cables, switches, and controllers.

Network file system

- NFS
- SMB

1.3. GFS2 FILE SYSTEM OVERVIEW

The Red Hat Global File System 2 (GFS2) file system is a 64-bit symmetric cluster file system which provides a shared name space and manages coherency between multiple nodes sharing a common block device. A GFS2 file system is intended to provide a feature set which is as close as possible to a local file system, while at the same time enforcing full cluster coherency between nodes. To achieve this, the nodes employ a cluster-wide locking scheme for file system resources. This locking scheme uses communication protocols such as TCP/IP to exchange locking information.

In a few cases, the Linux file system API does not allow the clustered nature of GFS2 to be totally transparent; for example, programs using POSIX locks in GFS2 should avoid using the **GETLK** function since, in a clustered environment, the process ID may be for a different node in the cluster. In most cases however, the functionality of a GFS2 file system is identical to that of a local file system.

The Red Hat Enterprise Linux Resilient Storage Add-On provides GFS2, and it depends on the Red Hat Enterprise Linux High Availability Add-On to provide the cluster management required by GFS2.

The **gfs2.ko** kernel module implements the GFS2 file system and is loaded on GFS2 cluster nodes.

To get the best performance from GFS2, it is important to take into account the performance considerations which stem from the underlying design. Just like a local file system, GFS2 relies on the page cache in order to improve performance by local caching of frequently used data. In order to maintain coherency across the nodes in the cluster, cache control is provided by the *glock* state machine.

Additional resources

- [Configuring GFS2 file systems](#)

1.4. GLUSTER STORAGE OVERVIEW

The Red Hat Gluster Storage (RHGS) is a software-defined storage platform that can be deployed in clusters. It aggregates disk storage resources from multiple servers into a single global namespace. GlusterFS is an open source distributed file system that is suitable for cloud and hybrid solutions.

Volumes form the base for GlusterFS and provide different requirements. Each volume is a collection of bricks, which are basic units of storage that are represented by an export directory on a server in the trusted storage pool.

The following types of GlusterFS volumes are available:

- **Distributed GlusterFS volume** is the default volume where each file is stored in one brick and the file cannot be shared between different bricks.
- **Replicated GlusterFS volume** type replicates user data, so that if one brick fails, the data is still accessible.
- **Distributed replicated GlusterFS volume** is a hybrid volume that distributes replicas over a large number of systems. It is suitable for environments where storage scalability and high-reliability are critical.

Additional resources

- [Red Hat gluster storage administration guide](#)

1.5. CEPH STORAGE OVERVIEW

Red Hat Ceph Storage (RHCS) is a scalable, open, software-defined storage platform that combines the most stable version of the Ceph storage system with a Ceph management platform, deployment utilities, and support services.

Red Hat Ceph Storage is designed for cloud infrastructure and web-scale object storage. Red Hat Ceph Storage clusters consist of the following types of nodes:

Red Hat Ceph Storage Ansible administration node

This type of node acts as the traditional Ceph Administration node did for previous versions of Red Hat Ceph Storage. This type of node provides the following functions:

- Centralized storage cluster management

- The Ceph configuration files and keys
- Optionally, local repositories for installing Ceph on nodes that cannot access the Internet for security reasons

Monitor nodes

Each monitor node runs the monitor daemon (**ceph-mon**), which maintains a copy of the cluster map. The cluster map includes the cluster topology. A client connecting to the Ceph cluster retrieves the current copy of the cluster map from the monitor which enables the client to read from and write data to the cluster.



IMPORTANT

Ceph can run with one monitor; however, to ensure high availability in a production cluster, Red Hat will only support deployments with at least three monitor nodes. Red Hat recommends deploying a total of 5 Ceph Monitors for storage clusters exceeding 750 OSDs.

OSD nodes

Each Object Storage Device (OSD) node runs the Ceph OSD daemon (**ceph-osd**), which interacts with logical disks attached to the node. Ceph stores data on these OSD nodes.

Ceph can run with very few OSD nodes, which the default is three, but production clusters realize better performance beginning at modest scales, for example 50 OSDs in a storage cluster. Ideally, a Ceph cluster has multiple OSD nodes, allowing isolated failure domains by creating the CRUSH map.

MDS nodes

Each Metadata Server (MDS) node runs the MDS daemon (**ceph-mds**), which manages metadata related to files stored on the Ceph File System (CephFS). The MDS daemon also coordinates access to the shared cluster.

Object Gateway node

Ceph Object Gateway node runs the Ceph RADOS Gateway daemon (**ceph-radosgw**), and is an object storage interface built on top of **librados** to provide applications with a RESTful gateway to Ceph Storage Clusters. The Ceph Object Gateway supports two interfaces:

S3

Provides object storage functionality with an interface that is compatible with a large subset of the Amazon S3 RESTful API.

Swift

Provides object storage functionality with an interface that is compatible with a large subset of the OpenStack Swift API.

Additional resources

- [Red Hat Ceph Storage](#)

CHAPTER 2. DISK PARTITIONS

To divide a disk into one or more logical areas, use the disk partitioning utility. It enables separate management of each partition.

2.1. OVERVIEW OF PARTITIONS

The hard disk stores information about the location and size of each disk partition in the partition table. Using information from the partition table, the operating system treats each partition as a logical disk. Some of the advantages of disk partitioning include:

- Reduce the likelihood of administrative oversights of Physical Volumes
- Ensure sufficient backup
- Provide efficient disk management

Additional resources

- [What are the advantages and disadvantages to using partitioning on LUNs, either directly or with LVM in between?](#)

2.2. CONSIDERATIONS BEFORE MODIFYING PARTITIONS ON A DISK

Before creating, removing, or resizing any disk partitions, consider the following aspects.

On a device, the type of the partition table determines the maximum number and size of individual partitions.

Maximum number of partitions:

- On a device formatted with the **Master Boot Record (MBR)** partition table, you can have:
 - Up to four primary partitions.
 - Up to three primary partitions, one extended partition
 - Multiple logical partitions within the extended partition
- On a device formatted with the **GUID Partition Table (GPT)** you can have:
 - Up to 128 partitions, if using the **parted** utility.
 - Though the GPT specification allows more partitions by increasing the reserved size of the partition table, the parted utility limits the area required for 128 partitions.

Maximum size of partitions:

- On a device formatted with the **Master Boot Record (MBR)** partition table, the maximum size is 2 TiB.
- On a device formatted with the **GUID Partition Table (GPT)** the maximum size is 8 ZiB.

By using the **parted** utility, you can specify the partition size using multiple different suffixes:

- **MiB, GiB, or TiB**

- Size expressed in powers of 2.
- The starting point of the partition is aligned to the exact sector specified by size.
- The ending point is aligned to the specified size minus 1 sector.
- **MB, GB, or TB:**
 - Size expressed in powers of 10.
 - The starting and ending points are aligned within one half of the specified unit. For example, $\pm 500\text{KB}$ when using the MB suffix.

**NOTE**

This section does not cover the DASD partition table, which is specific to the IBM Z architecture.

Additional resources

- [Configuring a Linux instance on IBM Z](#)
- [What you should know about DASD](#)

2.3. COMPARISON OF PARTITION TABLE TYPES

To enable partitions on a device, format a block device with different types of partition tables. The following table compares the properties of different types of partition tables that you can create on a block device.

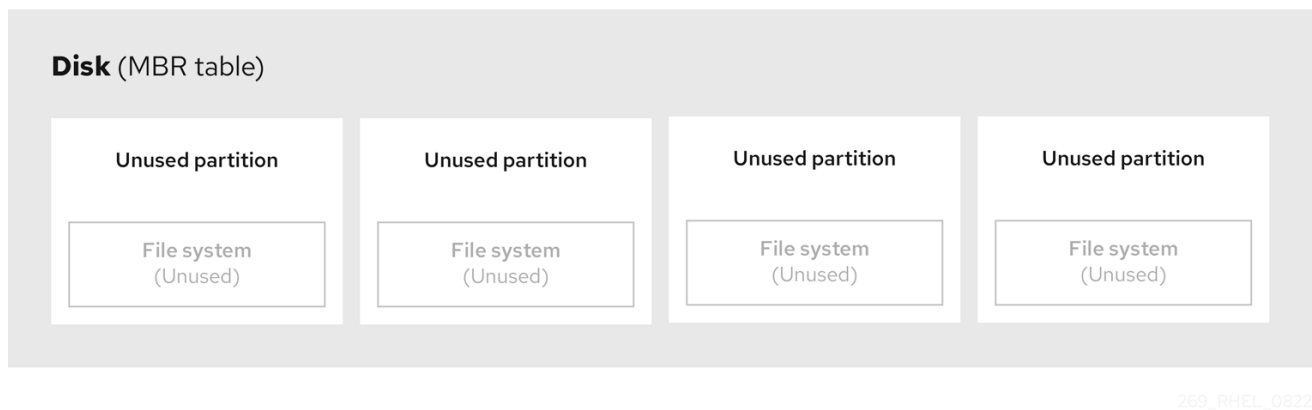
Table 2.1. Partition table types

Partition table	Maximum number of partitions	Maximum partition size
Master Boot Record (MBR)	4 primary, or 3 primary and 1 extended partition with 12 logical partitions	2TiB
GUID Partition Table (GPT)	128	8ZiB

2.4. MBR DISK PARTITIONS

The partition table is stored at the very start of the disk, before any file system or user data. For a more clear example, the partition table is shown as being separate in the following diagrams.

Figure 2.1. Disk with MBR partition table



As the previous diagram shows, the partition table is divided into four sections of four unused primary partitions. A primary partition is a partition on a hard drive that contains only one logical drive (or section). Each logical drive holds the information necessary to define a single partition, meaning that the partition table can define no more than four primary partitions.

Each partition table entry contains important characteristics of the partition:

- The points on the disk where the partition starts and ends
- The state of the partition, as only one partition can be flagged as **active**
- The type of partition

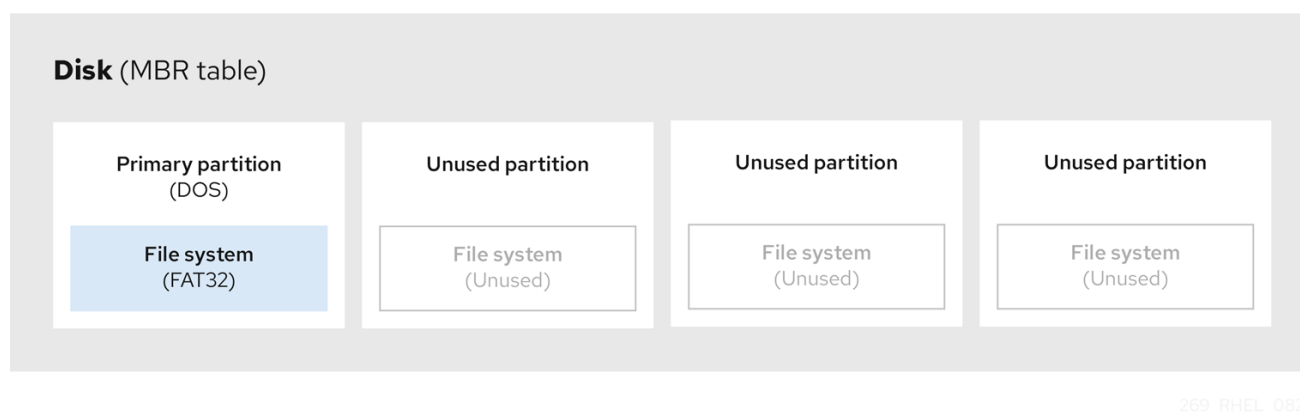
The starting and ending points define the size and location of the partition on the disk. Some of the operating systems boot loaders use the **active** flag. That means that the operating system in the partition that is marked "active" is booted.

The type is a number that identifies the anticipated usage of a partition. Some operating systems use the partition type to:

- Denote a specific file system type
- Flag the partition as being associated with a particular operating system
- Indicate that the partition contains a bootable operating system

The following diagram shows an example of a drive with a single partition. In this example, the first partition is labeled as **DOS** partition type:

Figure 2.2. Disk with a single partition



Additional resources

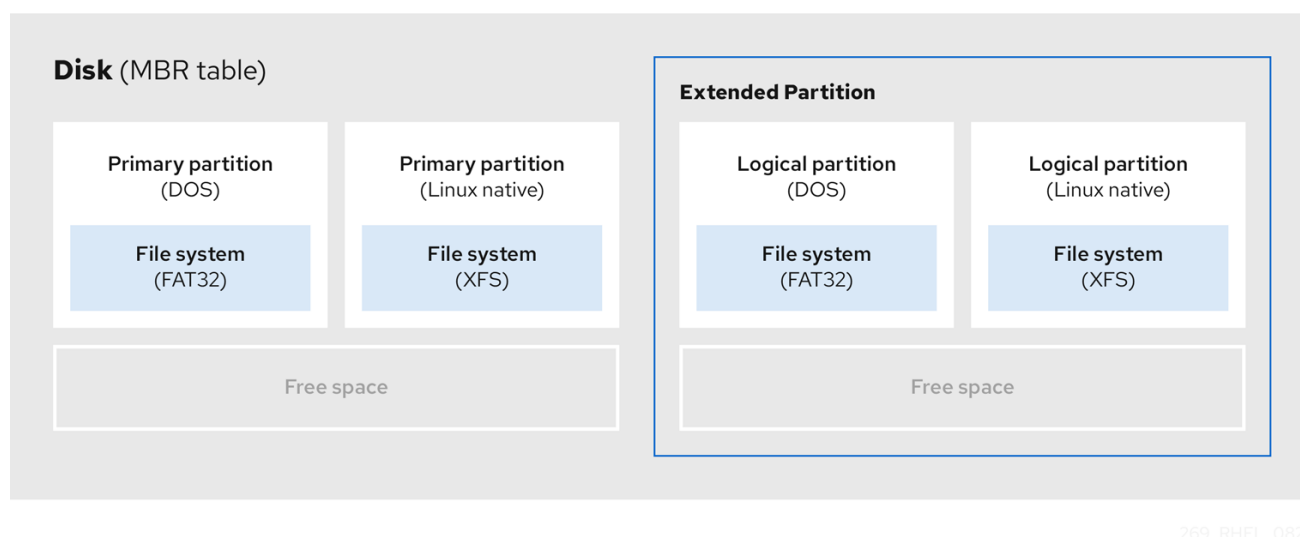
- [MBR partition types](#)

2.5. EXTENDED MBR PARTITIONS

To create additional partitions, if needed, set the type to **extended**.

An extended partition is similar to a disk drive. It has its own partition table, which points to one or more logical partitions, contained entirely within the extended partition. The following diagram shows a disk drive with two primary partitions, and one extended partition containing two logical partitions, along with some unpartitioned free space.

Figure 2.3. Disk with both two primary and an extended MBR partitions



You can have only up to four primary and extended partitions, but there is no fixed limit to the number of logical partitions. As a limit in Linux to access partitions, a single disk drive allows maximum 15 logical partitions.

2.6. MBR PARTITION TYPES

The table below shows a list of some of the most commonly used MBR partition types and hexadecimal numbers to represent them.

Table 2.2. MBR partition types

MBR partition type	Value	MBR partition type	Value
Empty	00	Novell Netware 386	65
DOS 12-bit FAT	01	PIC/IX	75
XENIX root	02	Old MINIX	80
XENIX usr	03	Linux/MINUX	81
DOS 16-bit ≤32M	04	Linux swap	82
Extended	05	Linux native	83
DOS 16-bit >=32	06	Linux extended	85
OS/2 HPFS	07	Amoeba	93
AIX	08	Amoeba BBT	94
AIX bootable	09	BSD/386	a5
OS/2 Boot Manager	0a	OpenBSD	a6
Win95 FAT32	0b	NEXTSTEP	a7
Win95 FAT32 (LBA)	0c	BSDI fs	b7
Win95 FAT16 (LBA)	0e	BSDI swap	b8
Win95 Extended (LBA)	0f	Syrinx	c7
Venix 80286	40	CP/M	db
Novell	51	DOS access	e1
PRep Boot	41	DOS R/O	e3
GNU HURD	63	DOS secondary	f2
Novell Netware 286	64	BBT	ff

2.7. GUID PARTITION TABLE

The GUID Partition Table (GPT) is a partitioning scheme based on the Globally Unique Identifier (GUID).

GPT deals with the limitations of the MBR partition table. The MBR partition table is unable to address storage larger than 2 TiB, equivalent to approximately 2.2 TB. Instead, GPT supports hard disks with larger capacity. The maximum addressable disk size is 2.2 ZiB. In addition, by default, GPT supports creation of up to 128 primary partitions. This number can be extended by allocating more space to the partition table.



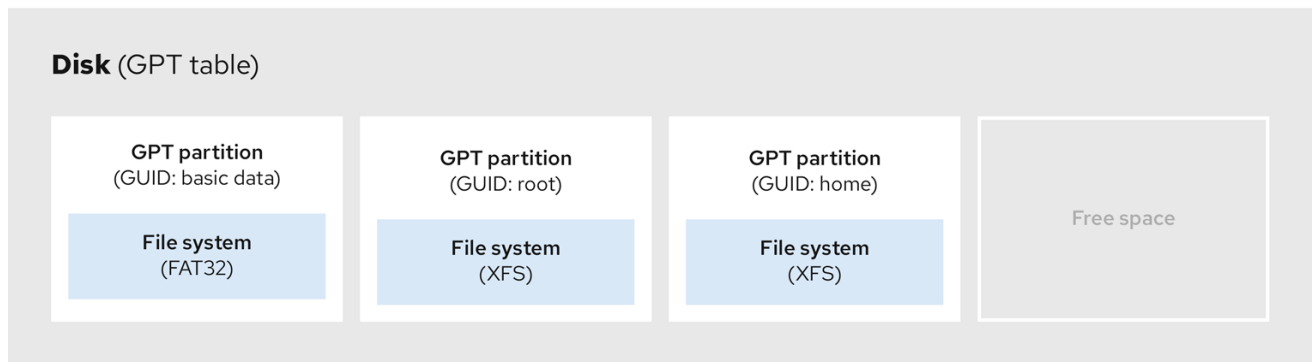
NOTE

A GPT has partition types based on GUIDs. Certain partitions require a specific GUID. For example, the system partition for EFI boot loaders require GUID **C12A7328-F81F-11D2-BA4B-00A0C93EC93B**.

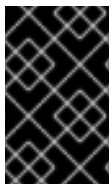
GPT disks use logical block addressing (LBA) and a partition layout as follows:

- For backward compatibility with MBR disks, the first sector (LBA 0) of GPT is reserved for MBR data and it is called "protective MBR".
- Primary GPT
 - The header begins on the second logical block (LBA 1) of the device. The header contains the disk GUID, the location of the primary partition table, the location of the secondary GPT header, and CRC32 checksums of itself, and the primary partition table. It also specifies the number of partition entries on the table.
 - By default, the primary GPT includes 128 partition entries. Each partition has an entry size of 128 bytes, a partition type GUID and a unique partition GUID.
- Secondary GPT
 - For recovery, it is useful as a backup table in case the primary partition table is corrupted.
 - The secondary GPT header is located on the last logical sector of the disk and recovers GPT information in case the primary header is corrupted.
 - It contains:
 - The disk GUID
 - The location of the secondary partition table and the primary GPT header
 - CRC32 checksums of itself
 - The secondary partition table
 - The number of possible partition entries

Figure 2.4. Disk with a GUID Partition Table



269_RHEL_0822



IMPORTANT

For a successful installation of the boot loader onto a GPT disk a BIOS boot partition must be present. If the disk already contains a BIOS boot partition, it can be reused. This includes disks initialized by the **Anaconda** installation program.

2.8. PARTITION TYPES

There are multiple ways to manage partition types:

- The **fdisk** utility supports the full range of partition types by specifying hexadecimal codes.
- The **systemd-gpt-auto-generator**, a unit generator utility, uses the partition type to automatically identify and mount devices.
- The **parted** utility maps out the partition type with *flags*. The **parted** utility handles only certain partition types, for example LVM, swap or RAID.

The **parted** utility supports setting the following flags:

- **boot**
- **root**
- **swap**
- **hidden**
- **raid**
- **lvm**
- **lba**
- **legacy_boot**
- **irst**
- **esp**
- **palo**

On Red Hat Enterprise Linux 9 with **parted** 3.5, you can use the additional flags **chromeos_kernel** and **bls_boot**.

The **parted** utility optionally accepts a file system type argument while creating a partition. See [Creating a partition with parted](#) for a list of the required conditions. Use the value to:

- Set the partition flags on MBR.
- Set the partition UUID type on GPT. For example, the **swap**, **fat**, or **hfs** file system types set different GUIDs. The default value is the Linux Data GUID.

The argument does not modify the file system on the partition. It only differentiates between the supported flags and GUIDs.

The following file system types are supported:

- **xfs**
- **ext2**
- **ext3**
- **ext4**
- **fat16**
- **fat32**
- **hfs**
- **hfs+**
- **linux-swap**
- **ntfs**
- **reiserfs**

2.9. PARTITION NAMING SCHEME

Red Hat Enterprise Linux uses a file-based naming scheme, with file names in the form of **/dev/xyN**.

Device and partition names consist of the following structure:

/dev/

Name of the directory that contains all device files. Hard disks contain partitions, thus the files representing all possible partitions are located in **/dev**.

xx

The first two letters of the partition name indicate the type of device that contains the partition.

y

This letter indicates the specific device containing the partition. For example, **/dev/sda** for the first hard disk and **/dev/sdb** for the second. You can use more letters in systems with more than 26 drives, for example, **/dev/sdaa1**.

N

The final letter indicates the number to represent the partition. The first four (primary or extended) partitions are numbered **1** through **4**. Logical partitions start at **5**. For example, **/dev/sda3** is the third primary or extended partition on the first hard disk, and **/dev/sdb6** is the second logical partition on the second hard disk. Drive partition numbering applies only to MBR partition tables. Note that **N** does not always mean partition.



NOTE

Even if Red Hat Enterprise Linux can identify and refer to *all* types of disk partitions, it might not be able to read the file system and therefore access stored data on every partition type. However, in many cases, it is possible to successfully access data on a partition dedicated to another operating system.

2.10. MOUNT POINTS AND DISK PARTITIONS

In Red Hat Enterprise Linux, each partition forms a part of the storage, necessary to support a single set of files and directories. Mounting a partition makes the storage of that partition available, starting at the specified directory known as a *mount point*.

For example, if partition **/dev/sda5** is mounted on **/usr/**, it means that all files and directories under **/usr/** physically reside on **/dev/sda5**. The file **/usr/share/doc/FAQ/txt/Linux-FAQ** resides on **/dev/sda5**, while the file **/etc/gdm/custom.conf** does not.

Continuing the example, it is also possible that one or more directories below **/usr/** would be mount points for other partitions. For example, **/usr/local/man/whatis** resides on **/dev/sda7**, rather than on **/dev/sda5**, if **/usr/local** includes a mounted **/dev/sda7** partition.

CHAPTER 3. GETTING STARTED WITH PARTITIONS

Use disk partitioning to divide a disk into one or more logical areas which enables work on each partition separately. The hard disk stores information about the location and size of each disk partition in the partition table. Using the table, each partition then appears as a logical disk to the operating system. You can then read and write on those individual disks.

For an overview of the advantages and disadvantages to using partitions on block devices, see [What are the advantages and disadvantages to using partitioning on LUNs, either directly or with LVM in between?](#).

3.1. CREATING A PARTITION TABLE ON A DISK WITH PARTED

Use the **parted** utility to format a block device with a partition table more easily.



WARNING

Formatting a block device with a partition table deletes all data stored on the device.

Procedure

1. Start the interactive **parted** shell:

```
# parted block-device
```

2. Determine if there already is a partition table on the device:

```
# (parted) print
```

If the device already contains partitions, they will be deleted in the following steps.

3. Create the new partition table:

```
# (parted) mklabel table-type
```

- Replace *table-type* with the intended partition table type:
 - **msdos** for MBR
 - **gpt** for GPT

Example 3.1. Creating a GUID Partition Table (GPT) table

To create a GPT table on the disk, use:

```
# (parted) mklabel gpt
```


The changes start applying after you enter this command.

4. View the partition table to confirm that it is created:

```
# (parted) print
```

5. Exit the **parted** shell:

```
# (parted) quit
```

Additional resources

- **parted(8)** man page.

3.2. VIEWING THE PARTITION TABLE WITH PARTED

Display the partition table of a block device to see the partition layout and details about individual partitions. You can view the partition table on a block device using the **parted** utility.

Procedure

1. Start the **parted** utility. For example, the following output lists the device **/dev/sda**:

```
# parted /dev/sda
```

2. View the partition table:

```
# (parted) print

Model: ATA SAMSUNG MZNLN256 (scsi)
Disk /dev/sda: 256GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number Start End Size Type File system Flags
 1    1049kB 269MB 268MB primary xfs      boot
 2    269MB 34.6GB 34.4GB primary
 3    34.6GB 45.4GB 10.7GB primary
 4    45.4GB 256GB 211GB extended
 5    45.4GB 256GB 211GB logical
```

3. Optional: Switch to the device you want to examine next:

```
# (parted) select block-device
```

For a detailed description of the print command output, see the following:

Model: ATA SAMSUNG MZNLN256 (scsi)

The disk type, manufacturer, model number, and interface.

Disk /dev/sda: 256GB

The file path to the block device and the storage capacity.

Partition Table: msdos

The disk label type.

Number

The partition number. For example, the partition with minor number 1 corresponds to **/dev/sda1**.

Start and End

The location on the device where the partition starts and ends.

Type

Valid types are metadata, free, primary, extended, or logical.

File system

The file system type. If the **File system** field of a device shows no value, this means that its file system type is unknown. The **parted** utility cannot recognize the file system on encrypted devices.

Flags

Lists the flags set for the partition. Available flags are **boot**, **root**, **swap**, **hidden**, **raid**, **lvm**, or **lba**.

Additional resources

- **parted(8)** man page.

3.3. CREATING A PARTITION WITH PARTED

As a system administrator, you can create new partitions on a disk by using the **parted** utility.

**NOTE**

The required partitions are **swap**, **/boot/**, and **/ (root)**.

Prerequisites

- A partition table on the disk.
- If the partition you want to create is larger than 2TiB, format the disk with the **GUID Partition Table (GPT)**.

Procedure

1. Start the **parted** utility:

```
# parted block-device
```

2. View the current partition table to determine if there is enough free space:

```
# (parted) print
```

- Resize the partition in case there is not enough free space.
- From the partition table, determine:
 - The start and end points of the new partition.
 - On MBR, what partition type it should be.

3. Create the new partition:

```
# (parted) mkpart part-type name fs-type start end
```

- Replace *part-type* with **primary**, **logical**, or **extended**. This applies only to the MBR partition table.
- Replace *name* with an arbitrary partition name. This is required for GPT partition tables.
- Replace *fs-type* with **xfs**, **ext2**, **ext3**, **ext4**, **fat16**, **fat32**, **hfs**, **hfs+**, **linux-swaps**, **ntfs**, or **reiserfs**. The *fs-type* parameter is optional. Note that the **parted** utility does not create the file system on the partition.
- Replace *start* and *end* with the sizes that determine the starting and ending points of the partition, counting from the beginning of the disk. You can use size suffixes, such as **512MiB**, **20GiB**, or **1.5TiB**. The default size is in megabytes.

Example 3.2. Creating a small primary partition

To create a primary partition from 1024MiB until 2048MiB on an MBR table, use:

```
# (parted) mkpart primary 1024MiB 2048MiB
```

The changes start applying after you enter the command.

4. View the partition table to confirm that the created partition is in the partition table with the correct partition type, file system type, and size:

```
# (parted) print
```

5. Exit the **parted** shell:

```
# (parted) quit
```

6. Register the new device node:

```
# udevadm settle
```

7. Verify that the kernel recognizes the new partition:

```
# cat /proc/partitions
```

Additional resources

- **parted(8)** man page.
- [Creating a partition table on a disk with parted](#)
- [Resizing a partition with parted](#)

3.4. SETTING A PARTITION TYPE WITH FDISK

You can set a partition type or flag, using the **fdisk** utility.

Prerequisites

- A partition on the disk.

Procedure

1. Start the interactive **fdisk** shell:

```
# fdisk block-device
```

2. View the current partition table to determine the minor partition number:

```
Command (m for help): print
```

You can see the current partition type in the **Type** column and its corresponding type ID in the **Id** column.

3. Enter the partition type command and select a partition using its minor number:

```
Command (m for help): type  
Partition number (1,2,3 default 3): 2
```

4. Optional: View the list in hexadecimal codes:

```
Hex code (type L to list all codes): L
```

5. Set the partition type:

```
Hex code (type L to list all codes): 8e
```

6. Write your changes and exit the **fdisk** shell:

```
Command (m for help): write  
The partition table has been altered.  
Syncing disks.
```

7. Verify your changes:

```
# fdisk --list block-device
```

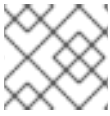
3.5. RESIZING A PARTITION WITH PARTED

Using the **parted** utility, extend a partition to utilize unused disk space, or shrink a partition to use its capacity for different purposes.

Prerequisites

- Back up the data before shrinking a partition.

- If the partition you want to create is larger than 2TiB, format the disk with the **GUID Partition Table (GPT)**.
- If you want to shrink the partition, first shrink the file system so that it is not larger than the resized partition.

**NOTE**

XFS does not support shrinking.

Procedure

1. Start the **parted** utility:

```
# parted block-device
```

2. View the current partition table:

```
# (parted) print
```

From the partition table, determine:

- The minor number of the partition.
- The location of the existing partition and its new ending point after resizing.

3. Resize the partition:

```
# (parted) resizepart 1 2GiB
```

- Replace *1* with the minor number of the partition that you are resizing.
- Replace *2* with the size that determines the new ending point of the resized partition, counting from the beginning of the disk. You can use size suffixes, such as **512MiB**, **20GiB**, or **1.5TiB**. The default size is in megabytes.

4. View the partition table to confirm that the resized partition is in the partition table with the correct size:

```
# (parted) print
```

5. Exit the **parted** shell:

```
# (parted) quit
```

6. Verify that the kernel registers the new partition:

```
# cat /proc/partitions
```

7. Optional: If you extended the partition, extend the file system on it as well.

Additional resources

- **parted(8)** man page.

- [Creating a partition table on a disk with parted](#)
- [Resizing an ext3 file system](#)
- [Increasing the size of an XFS file system](#)

3.6. REMOVING A PARTITION WITH PARTED

Using the **parted** utility, you can remove a disk partition to free up disk space.



WARNING

Removing a partition deletes all data stored on the partition.

Procedure

1. Start the interactive **parted** shell:

```
# parted block-device
```

2. View the current partition table to determine the minor number of the partition to remove:

```
# (parted) print
```

3. Remove the partition:

```
# (parted) rm minor-number
```

- Replace *minor-number* with the minor number of the partition you want to remove.

The changes start applying as soon as you enter this command.

4. Verify that you have removed the partition from the partition table:

```
# (parted) print
```

5. Exit the **parted** shell:

```
# (parted) quit
```

6. Verify that the kernel registers that the partition is removed:

```
# cat /proc/partitions
```

7. Remove the partition from the **/etc/fstab** file, if it is present. Find the line that declares the removed partition, and remove it from the file.

8. Regenerate mount units so that your system registers the new **/etc/fstab** configuration:

—

```
# systemctl daemon-reload
```

9. If you have deleted a swap partition or removed pieces of LVM, remove all references to the partition from the kernel command line in the `/etc/default/grub` file, and regenerate GRUB configuration:

- On a BIOS-based system:

```
# grub2-mkconfig --output=/etc/grub2.cfg
```

- On a UEFI-based system:

```
# grub2-mkconfig --output=/etc/grub2-efi.cfg
```

10. To register the changes in the early boot system, rebuild the **initramfs** file system:

```
# dracut --force --verbose
```

Additional resources

- **parted(8)** man page

CHAPTER 4. STRATEGIES FOR REPARTITIONING A DISK

There are different approaches to repartitioning a disk. These include:

- Unpartitioned free space is available.
- An unused partition is available.
- Free space in an actively used partition is available.



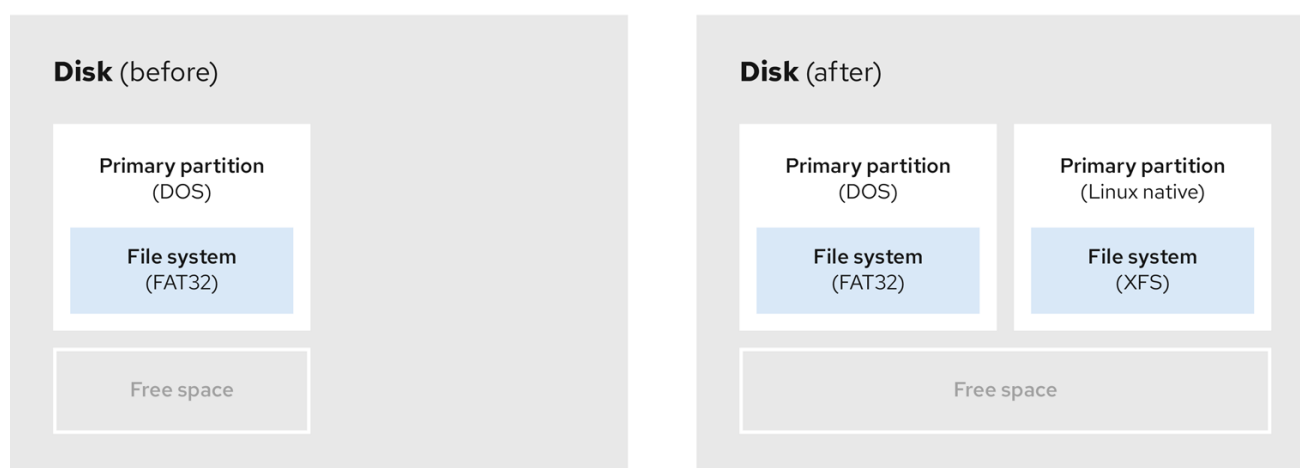
NOTE

The following examples are simplified for clarity and do not reflect the exact partition layout when actually installing Red Hat Enterprise Linux.

4.1. USING UNPARTITIONED FREE SPACE

Partitions that are already defined and do not span the entire hard disk, leave unallocated space that is not part of any defined partition. The following diagram shows what this might look like.

Figure 4.1. Disk with unpartitioned free space



269_RHEL_0822

The first diagram represents a disk with one primary partition and an undefined partition with unallocated space. The second diagram represents a disk with two defined partitions with allocated space.

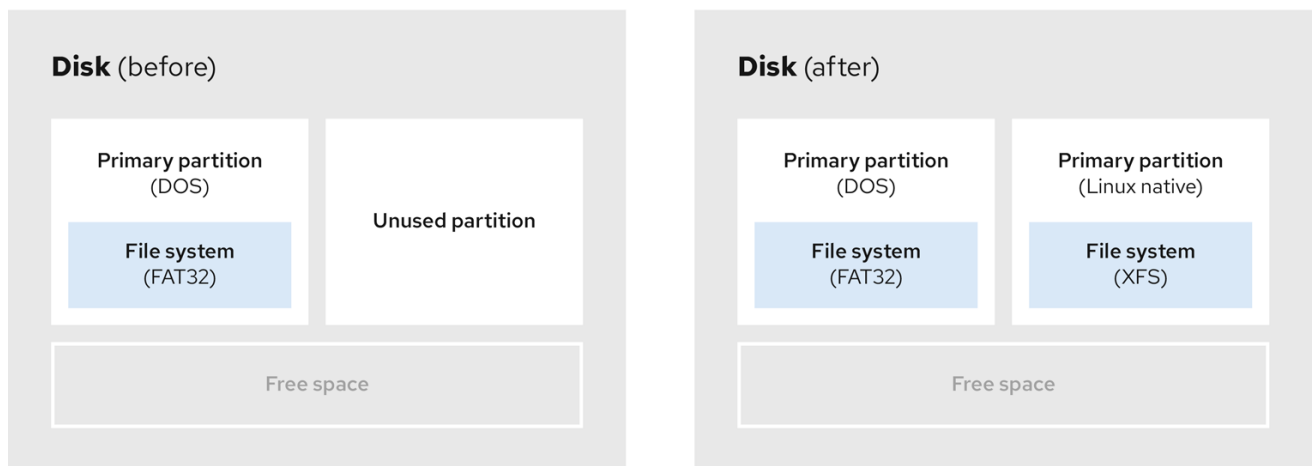
An unused hard disk also falls into this category. The only difference is that *all* the space is not part of any defined partition.

On a new disk, you can create the necessary partitions from the unused space. Most preinstalled operating systems are configured to take up all available space on a disk drive.

4.2. USING SPACE FROM AN UNUSED PARTITION

In the following example, the first diagram represents a disk with an unused partition. The second diagram represents reallocating an unused partition for Linux.

Figure 4.2. Disk with an unused partition



269_RHEL_0822

To use the space allocated to the unused partition, delete the partition and then create the appropriate Linux partition instead. Alternatively, during the installation process, delete the unused partition and manually create new partitions.

4.3. USING FREE SPACE FROM AN ACTIVE PARTITION

This process can be difficult to manage because an active partition, that is already in use, contains the required free space. In most cases, hard disks of computers with preinstalled software contain one larger partition holding the operating system and data.



WARNING

If you want to use an operating system (OS) on an active partition, you must reinstall the OS. Be aware that some computers, which include pre-installed software, do not include installation media to reinstall the original OS. Check whether this applies to your OS before you destroy an original partition and the OS installation.

To optimise the use of available free space, you can use the methods of destructive or non-destructive repartitioning.

4.3.1. Destructive repartitioning

Destructive repartitioning destroys the partition on your hard drive and creates several smaller partitions instead. Backup any needed data from the original partition as this method deletes the complete contents.

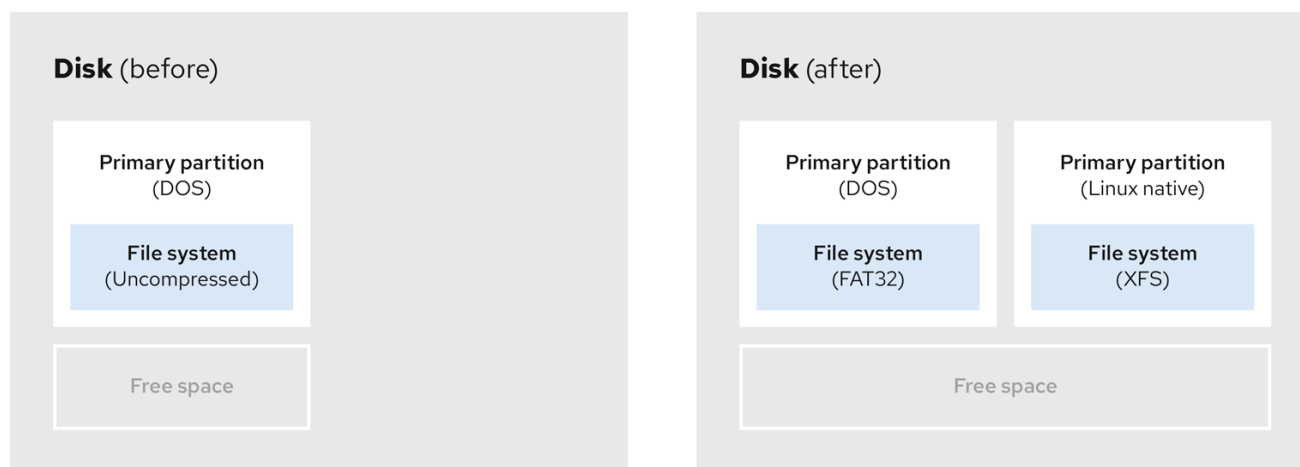
After creating a smaller partition for your existing operating system, you can:

- Reinstall software.
- Restore your data.

- Start your Red Hat Enterprise Linux installation.

The following diagram is a simplified representation of using the destructive repartitioning method.

Figure 4.3. Destructive repartitioning action on disk



269_RHEL_0822



WARNING

This method deletes all data previously stored in the original partition.

4.3.2. Non-destructive repartitioning

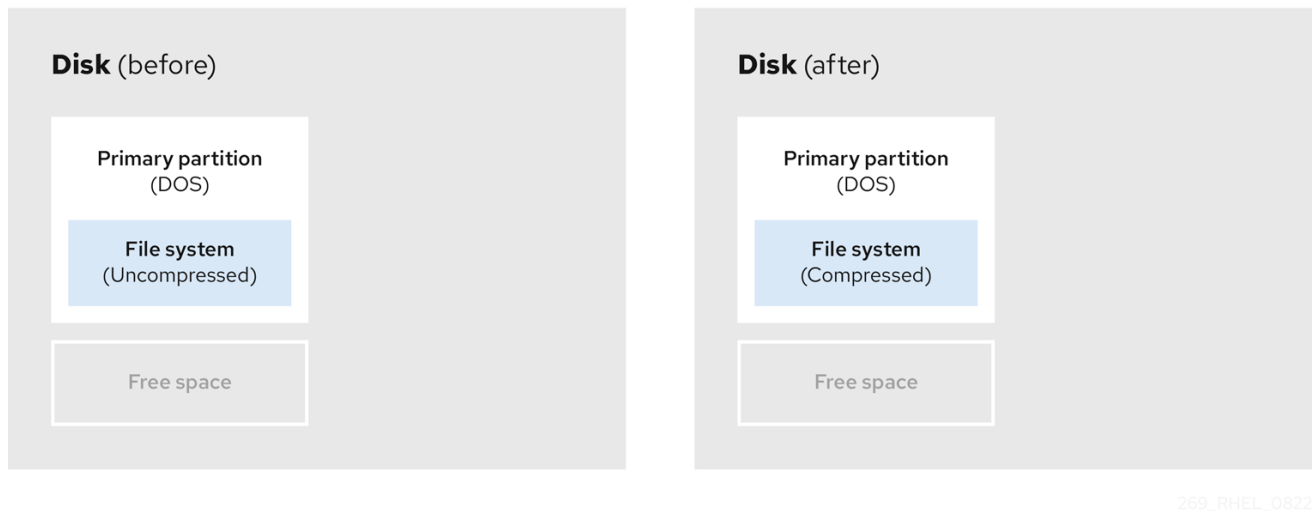
Non-destructive repartitioning resizes partitions, without any data loss. This method is reliable, however it takes longer processing time on large drives.

The following is a list of methods, which can help initiate non-destructive repartitioning.

- Compress existing data

The storage location of some data cannot be changed. This can prevent the resizing of a partition to the required size, and ultimately lead to a destructive repartition process. Compressing data in an already existing partition can help you resize your partitions as needed. It can also help to maximize the free space available.

The following diagram is a simplified representation of this process.

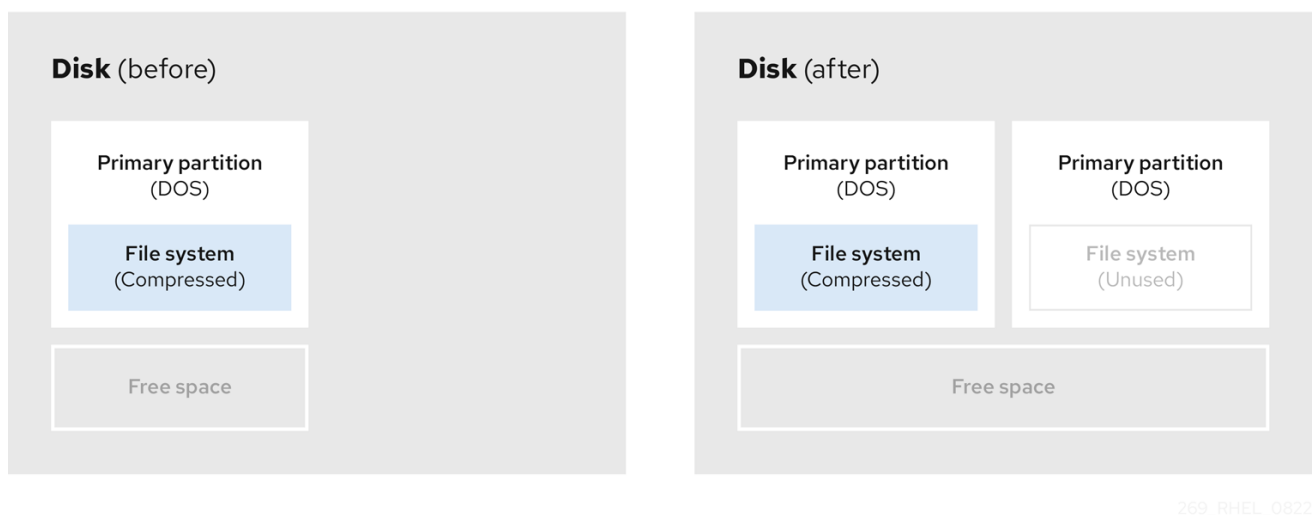
Figure 4.4. Data compression on a disk

To avoid any possible data loss, create a backup before continuing with the compression process.

- Resize the existing partition

By resizing an already existing partition, you can free up more space. Depending on your resizing software, the results may vary. In the majority of cases, you can create a new unformatted partition of the same type, as the original partition.

The steps you take after resizing can depend on the software you use. In the following example, the best practice is to delete the new DOS (Disk Operating System) partition, and create a Linux partition instead. Verify what is most suitable for your disk before initiating the resizing process.

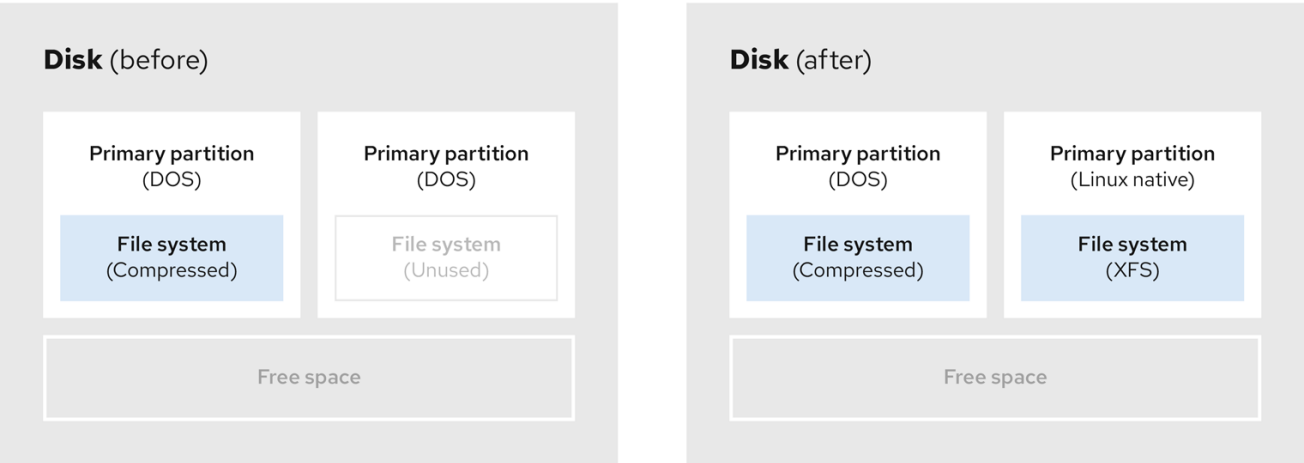
Figure 4.5. Partition resizing on a disk

- Optional: Create new partitions

Some pieces of resizing software support Linux based systems. In such cases, there is no need to delete the newly created partition after resizing. Creating a new partition afterwards depends on the software you use.

The following diagram represents the disk state, before and after creating a new partition.

Figure 4.6. Disk with final partition configuration



269_RHEL_0822

CHAPTER 5. CONFIGURING AN ISCSI TARGET

Red Hat Enterprise Linux uses the **targetcli** shell as a command-line interface to perform the following operations:

- Add, remove, view, and monitor iSCSI storage interconnects to utilize iSCSI hardware.
- Export local storage resources that are backed by either files, volumes, local SCSI devices, or by RAM disks to remote systems.

The **targetcli** tool has a tree-based layout including built-in tab completion, auto-complete support, and inline documentation.

5.1. INSTALLING TARGETCLI

Install the **targetcli** tool to add, monitor, and remove iSCSI storage interconnects .

Procedure

1. Install the **targetcli** tool:

```
# dnf install targetcli
```

2. Start the target service:

```
# systemctl start target
```

3. Configure target to start at boot time:

```
# systemctl enable target
```

4. Open port **3260** in the firewall and reload the firewall configuration:

```
# firewall-cmd --permanent --add-port=3260/tcp
Success
```

```
# firewall-cmd --reload
Success
```

Verification

- View the **targetcli** layout:

```
# targetcli
/> 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]
```

Additional resources

- **targetcli(8)** man page

5.2. CREATING AN ISCSI TARGET

Creating an iSCSI target enables the iSCSI initiator of the client to access the storage devices on the server. Both targets and initiators have unique identifying names.

Prerequisites

- Installed and running **targetcli**. For more information, see [Installing targetcli](#).

Procedure

1. Navigate to the iSCSI directory:

```
/ > iscsi/
```



NOTE

The **cd** command is used to change directories as well as to list the path to move into.

2. Use one of the following options to create an iSCSI target:

- a. Creating an iSCSI target using a default target name:

```
/iscsi> create
```

```
Created target
iqn.2003-01.org.linux-iscsi.hostname.x8664:sn.78b473f296ff
Created TPG1
```

- b. Creating an iSCSI target using a specific name:

```
/iscsi> create iqn.2006-04.com.example:444
```

```
Created target iqn.2006-04.com.example:444
Created TPG1
Here iqn.2006-04.com.example:444 is target_iqn_name
```

Replace *iqn.2006-04.com.example:444* with the specific target name.

3. Verify the newly created target:

```
/iscsi> ls
```

```
o- iscsi.....[1 Target]
  o- iqn.2006-04.com.example:444.....[1 TPG]
    o- tpg1.....[enabled, auth]
```

```
o- acls.....[0 ACL]
o- luns.....[0 LUN]
o- portals.....[0 Portal]
```

Additional resources

- **targetcli(8)** man page

5.3. ISCSI BACKSTORE

An iSCSI backstore enables support for different methods of storing an exported LUN's data on the local machine. Creating a storage object defines the resources that the backstore uses.

An administrator can choose any of the following backstore devices that Linux-IO (LIO) supports:

fileio backstore

Create a **fileio** storage object if you are using regular files on the local file system as disk images. For creating a **fileio** backstore, see [Creating a fileio storage object](#).

block backstore

Create a **block** storage object if you are using any local block device and logical device. For creating a **block** backstore, see [Creating a block storage object](#).

pscsi backstore

Create a **pscsi** storage object if your storage object supports direct pass-through of SCSI commands. For creating a **pscsi** backstore, see [Creating a pscsi storage object](#).

ramdisk backstore

Create a **ramdisk** storage object if you want to create a temporary RAM backed device. For creating a **ramdisk** backstore, see [Creating a Memory Copy RAM disk storage object](#).

Additional resources

- **targetcli(8)** man page

5.4. CREATING A FILEIO STORAGE OBJECT

fileio storage objects can support either the **write_back** or **write_thru** operations. The **write_back** operation enables the local file system cache. This improves performance but increases the risk of data loss.

It is recommended to use **write_back=false** to disable the **write_back** operation in favor of the **write_thru** operation.

Prerequisites

- Installed and running **targetcli**. For more information, see [Installing targetcli](#).

Procedure

1. Navigate to the **fileio/** from the **backstores/** directory:

```
/> backstores/fileio
```

2. Create a **fileio** storage object:

```
/backstores/fileio> create file1 /tmp/disk1.img 200M write_back=false  
  
Created fileio file1 with size 209715200
```

Verification

- Verify the created **fileio** storage object:

```
/backstores/fileio> ls
```

Additional resources

- **targetcli(8)** man page

5.5. CREATING A BLOCK STORAGE OBJECT

The block driver allows the use of any block device that appears in the **/sys/block/** directory to be used with Linux-IO (LIO). This includes physical devices such as, HDDs, SSDs, CDs, and DVDs, and logical devices such as, software or hardware RAID volumes, or LVM volumes.

Prerequisites

- Installed and running **targetcli**. For more information, see [Installing targetcli](#).

Procedure

1. Navigate to the **block/** from the **backstores/** directory:

```
/> backstores/block/
```

2. Create a **block** backstore:

```
/backstores/block> create name=block_backend dev=/dev/sdb  
  
Generating a wwn serial.  
Created block storage object block_backend using /dev/vdb.
```

Verification

- Verify the created **block** storage object:

```
/backstores/block> ls
```



NOTE

You can also create a **block** backstore on a logical volume.

Additional resources

- **targetcli(8)** man page

5.6. CREATING A PSCSI STORAGE OBJECT

You can configure, as a backstore, any storage object that supports direct pass-through of SCSI commands without SCSI emulation, and with an underlying SCSI device that appears with **lsscsi** in the **/proc/scsi/scsi** such as, a SAS hard drive . SCSI-3 and higher is supported with this subsystem.



WARNING

pscsi should only be used by advanced users. Advanced SCSI commands such as for Asymmetric Logical Unit Assignment (ALUAs) or Persistent Reservations (for example, those used by VMware ESX, and vSphere) are usually not implemented in the device firmware and can cause malfunctions or crashes. When in doubt, use **block** backstore for production setups instead.

Prerequisites

- Installed and running **targetcli**. For more information, see [Installing targetcli](#).

Procedure

1. Navigate to the **pscsi/** from the **backstores/** directory:

```
/ > backstores/pscsi/
```

2. Create a **pscsi** backstore for a physical SCSI device, a TYPE_ROM device using **/dev/sr0** in this example:

```
/backstores/pscsi> create name=pscsi_backend dev=/dev/sr0
```

```
Generating a wwn serial.
```

```
Created pscsi storage object pscsi_backend using /dev/sr0
```

Verification

- Verify the created **pscsi** storage object:

```
/backstores/pscsi> ls
```

Additional resources

- **targetcli(8)** man page

5.7. CREATING A MEMORY COPY RAM DISK STORAGE OBJECT

Memory Copy RAM disks (**ramdisk**) provide RAM disks with full SCSI emulation and separate memory mappings using memory copy for initiators. This provides capability for multi-sessions and is particularly useful for fast and volatile mass storage for production purposes.

Prerequisites

- Installed and running **targetcli**. For more information, see [Installing targetcli](#).

Procedure

1. Navigate to the **ramdisk/** from the **backstores/** directory:

```
/ > backstores/ramdisk/
```

2. Create a 1GB RAM disk backstore:

```
/backstores/ramdisk> create name=rd_backend size=1GB
```

Generating a wwn serial.

Created rd_mcp ramdisk rd_backend with size 1GB.

Verification

- Verify the created **ramdisk** storage object:

```
/backstores/ramdisk> ls
```

Additional resources

- **targetcli(8)** man page

5.8. CREATING AN ISCSI PORTAL

Creating an iSCSI portal adds an IP address and a port to the target that keeps the target enabled.

Prerequisites

- Installed and running **targetcli**. For more information, see [Installing targetcli](#).
- An iSCSI target associated with a Target Portal Groups (TPG). For more information, see [Creating an iSCSI target](#).

Procedure

1. Navigate to the TPG directory:

```
/iscsi> iqn.2006-04.example:444/tpg1/
```

2. Use one of the following options to create an iSCSI portal:

- a. Creating a default portal uses the default iSCSI port **3260** and allows the target to listen to all IP addresses on that port:

```
/iscsi/iqn.20...mple:444/tpg1> portals/ create
```

```
Using default IP port 3260
Binding to INADDR_Any (0.0.0.0)
Created network portal 0.0.0.0:3260
```



NOTE

When an iSCSI target is created, a default portal is also created. This portal is set to listen to all IP addresses with the default port number that is:
0.0.0.0:3260.

To remove the default portal, use the following command:

```
/iscsi/iqn-name/tpg1/portals delete ip_address=0.0.0.0 ip_port=3260
```

- b. Creating a portal using a specific IP address:

```
/iscsi/iqn.20...mple:444/tpg1> portals/ create 192.168.122.137
```

```
Using default IP port 3260
Created network portal 192.168.122.137:3260
```

Verification

- Verify the newly created portal:

```
/iscsi/iqn.20...mple:444/tpg1> ls
o- tpg..... [enambled, auth]
  o- acls .....[0 ACL]
  o- luns .....[0 LUN]
  o- portals .....[1 Portal]
    o- 192.168.122.137:3260.....[OK]
```

Additional resources

- **targetcli(8)** man page

5.9. CREATING AN ISCSI LUN

Logical unit number (LUN) is a physical device that is backed by the iSCSI backstore. Each LUN has a unique number.

Prerequisites

- Installed and running **targetcli**. For more information, see [Installing targetcli](#).
- An iSCSI target associated with a Target Portal Groups (TPG). For more information, see [Creating an iSCSI target](#).
- Created storage objects. For more information, see [iSCSI Backstore](#).

Procedure

1. Create LUNs of already created storage objects:

```
/iscsi/iqn.20...mple:444/tpg1> luns/ create /backstores/ramdisk/rd_backend
Created LUN 0.
```

```
/iscsi/iqn.20...mple:444/tpg1> luns/ create /backstores/block/block_backend
Created LUN 1.
```

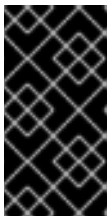
```
/iscsi/iqn.20...mple:444/tpg1> luns/ create /backstores/fileio/file1
Created LUN 2.
```

2. Verify the created LUNs:

```
/iscsi/iqn.20...mple:444/tpg1> ls

o- tpg..... [enambled, auth]
  o- acls .....[0 ACL]
  o- luns .....[3 LUNs]
    | o- lun0.....[ramdisk/ramdisk1]
    | o- lun1.....[block/block1 (/dev/vdb1)]
    | o- lun2.....[fileio/file1 (/foo.img)]
  o- portals .....[1 Portal]
    o- 192.168.122.137:3260.....[OK]
```

Default LUN name starts at **0**.



IMPORTANT

By default, LUNs are created with read-write permissions. If a new LUN is added after ACLs are created, LUN automatically maps to all available ACLs and can cause a security risk. To create a LUN with read-only permissions, see [Creating a read-only iSCSI LUN](#).

3. Configure ACLs. For more information, see [Creating an iSCSI ACL](#).

Additional resources

- **targetcli(8)** man page

5.10. CREATING A READ-ONLY ISCSI LUN

By default, LUNs are created with read-write permissions. This procedure describes how to create a read-only LUN.

Prerequisites

- Installed and running **targetcli**. For more information, see [Installing targetcli](#).
- An iSCSI target associated with a Target Portal Groups (TPG). For more information, see [Creating an iSCSI target](#).
- Created storage objects. For more information, see [iSCSI Backstore](#).

Procedure

1. Set read-only permissions:

```
/> set global auto_add_mapped_luns=false
```

Parameter `auto_add_mapped_luns` is now 'false'.

This prevents the auto mapping of LUNs to existing ACLs allowing the manual mapping of LUNs.

2. Navigate to the `initiator_iqn_name` directory:

```
/> iscsi/target_iqn_name/tpg1/acls/initiator_iqn_name/
```

3. Create the LUN:

```
/iscsi/target_iqn_name/tpg1/acls/initiator_iqn_name> create  
mapped_lun=next_sequential_LUN_number tpg_lun_or_backstore=backstore  
write_protect=1
```

Example:

```
/iscsi/target_iqn_name/tpg1/acls/2006-04.com.example.foo:888> create mapped_lun=1  
tpg_lun_or_backstore=/backstores/block/block2 write_protect=1
```

Created LUN 1.
Created Mapped LUN 1.

4. Verify the created LUN:

```
/iscsi/target_iqn_name/tpg1/acls/2006-04.com.example.foo:888> ls  
o- 2006-04.com.example.foo:888 .. [Mapped LUNs: 2]  
| o- mapped_lun0 ..... [lun0 block/disk1 (rw)]  
| o- mapped_lun1 ..... [lun1 block/disk2 (ro)]
```

The `mapped_lun1` line now has (**ro**) at the end (unlike `mapped_lun0`'s (**rw**)) stating that it is read-only.

5. Configure ACLs. For more information, see [Creating an iSCSI ACL](#).

Additional resources

- **targetcli(8)** man page

5.11. CREATING AN ISCSI ACL

In **targetcli**, Access Control Lists (ACLs) are used to define access rules and each initiator has exclusive access to a LUN.

Both targets and initiators have unique identifying names. You must know the unique name of the initiator to configure ACLs. The iSCSI initiators can be found in the `/etc/iscsi/initiatorname.iscsi` file.

Prerequisites

- Installed and running **targetcli**. For more information, see [Installing targetcli](#).
- An iSCSI target associated with a Target Portal Groups (TPG). For more information, see [Creating an iSCSI target](#).

Procedure

1. Navigate to the acls directory

```
/iscsi/iqn.20...mple:444/tpg1> acls/
```

2. Use one of the following options to create an ACL :
 - a. Using the initiator name from **/etc/iscsi/initiatorname.iscsi** file on the initiator.
 - b. Using a name that is easier to remember, see section [Creating an iSCSI initiator](#) to ensure ACL matches the initiator.

```
/iscsi/iqn.20...444/tpg1/acls> create iqn.2006-04.com.example.foo:888
```

```
Created Node ACL for iqn.2006-04.com.example.foo:888
Created mapped LUN 2.
Created mapped LUN 1.
Created mapped LUN 0.
```



NOTE

The global setting **auto_add_mapped_luns** used in the preceding example, automatically maps LUNs to any created ACL.

You can set user-created ACLs within the TPG node on the target server:

```
/iscsi/iqn.20...scsi:444/tpg1> set attribute generate_node_acls=1
```

Verification

- Verify the created ACL:

```
/iscsi/iqn.20...444/tpg1/acls> ls

o- acls .....[1 ACL]
  o- iqn.2006-04.com.example.foo:888 ....[3 Mapped LUNs, auth]
    o- mapped_lun0 .....[lun0 ramdisk/ramdisk1 (rw)]
    o- mapped_lun1 .....[lun1 block/block1 (rw)]
    o- mapped_lun2 .....[lun2 fileio/file1 (rw)]
```

Additional resources

- **targetcli(8)** man page

5.12. SETTING UP THE CHALLENGE-HANDSHAKE AUTHENTICATION PROTOCOL FOR THE TARGET

By using the **Challenge-Handshake Authentication Protocol (CHAP)**, users can protect the target with a password. The initiator must be aware of this password to be able to connect to the target.

Prerequisites

- Created iSCSI ACL. For more information, see [Creating an iSCSI ACL](#).

Procedure

1. Set attribute authentication:

```
/iscsi/iqn.20...mple:444/tpg1> set attribute authentication=1
Parameter authentication is now '1'.
```

2. Set **userid** and **password**:

```
/tpg1> set auth userid=redhat
Parameter userid is now 'redhat'.

/iscsi/iqn.20...689dcbb3/tpg1> set auth password=redhat_passwd
Parameter password is now 'redhat_passwd'.
```

Additional resources

- **targetcli(8)** man page

5.13. REMOVING AN ISCSI OBJECT USING TARGETCLI TOOL

This procedure describes how to remove the iSCSI objects using the **targetcli** tool.

Procedure

1. Log off from the target:

```
# iscsiadm -m node -T iqn.2006-04.example:444 -u
```

For more information on how to log in to the target, see [Creating an iSCSI initiator](#).

2. Remove the entire target, including all ACLs, LUNs, and portals:

```
/> iscsi/ delete iqn.2006-04.com.example:444
```

Replace *iqn.2006-04.com.example:444* with the `target_iqn_name`.

- To remove an iSCSI backstore:

```
/> backstores/backstore-type/ delete block_backend
```

- Replace *backstore-type* with either **fileio**, **block**, **pscsi**, or **ramdisk**.

- Replace *block_backend* with the *backstore-name* you want to delete.
- To remove parts of an iSCSI target, such as an ACL:

```
/> /iscsi/iqn-name/tpg/acls/ delete iqn.2006-04.com.example:444
```

Verification

- View the changes:

```
/> iscsi/ ls
```

Additional resources

- **targetcli(8)** man page

CHAPTER 6. CONFIGURING AN ISCSI INITIATOR

An iSCSI initiator forms a session to connect to the iSCSI target. By default, an iSCSI service is lazily started and the service starts after running the **iscsiadm** command. If root is not on an iSCSI device or there are no nodes marked with **node.startup = automatic** then the iSCSI service will not start until an **iscsiadm** command is executed that requires **iscsid** or the **iscsi** kernel modules to be started.

Execute the **systemctl start iscsid.service** command as root to force the **iscsid** daemon to run and iSCSI kernel modules to load.

6.1. CREATING AN ISCSI INITIATOR

This section describes how to create an iSCSI initiator.

Prerequisites

- Installed and running **targetcli** on a server machine. For more information, see [Installing targetcli](#).
- An iSCSI target associated with a Target Portal Groups (TPG) on a server machine. For more information, see [Creating an iSCSI target](#).
- Created iSCSI ACL. For more information, see [Creating an iSCSI ACL](#).

Procedure

1. Install **iscsi-initiator-utils** on client machine:

```
# dnf install iscsi-initiator-utils
```

2. Check the initiator name:

```
# cat /etc/iscsi/initiatorname.iscsi

InitiatorName=2006-04.com.example.foo:888
```

3. If the ACL was given a custom name in [Creating an iSCSI ACL](#), modify the **/etc/iscsi/initiatorname.iscsi** file accordingly.

```
# vi /etc/iscsi/initiatorname.iscsi
```

4. Discover the target and log in to the target with the displayed target IQN:

```
# iscsiadm -m discovery -t st -p 10.64.24.179
10.64.24.179:3260,1 iqn.2006-04.example:444

# iscsiadm -m node -T iqn.2006-04.example:444 -l
Logging in to [iface: default, target: iqn.2006-04.example:444, portal: 10.64.24.179,3260]
(multiple)
Login to [iface: default, target: iqn.2006-04.example:444, portal: 10.64.24.179,3260]
successful.
```

Replace *10.64.24.179* with the target-ip-address.

You can use this procedure for any number of initiators connected to the same target if their respective initiator names are added to the ACL as described in the [Creating an iSCSI ACL](#).

5. Find the iSCSI disk name and create a file system on this iSCSI disk:

```
# grep "Attached SCSI" /var/log/messages  
  
# mkfs.ext4 /dev/disk_name
```

Replace *disk_name* with the iSCSI disk name displayed in the `/var/log/messages` file.

6. Mount the file system:

```
# mkdir /mount/point  
  
# mount /dev/disk_name /mount/point
```

Replace `/mount/point` with the mount point of the partition.

7. Edit the `/etc/fstab` file to mount the file system automatically when the system boots:

```
# vi /etc/fstab  
  
/dev/disk_name /mount/point ext4 _netdev 0 0
```

Replace *disk_name* with the iSCSI disk name and `/mount/point` with the mount point of the partition.

Additional resources

- **targetcli(8)** and **iscsiadm(8)** man pages

6.2. SETTING UP THE CHALLENGE-HANDSHAKE AUTHENTICATION PROTOCOL FOR THE INITIATOR

By using the **Challenge-Handshake Authentication Protocol (CHAP)**, users can protect the target with a password. The initiator must be aware of this password to be able to connect to the target.

Prerequisites

- Created iSCSI initiator. For more information, see [Creating an iSCSI initiator](#).
- Set the **CHAP** for the target. For more information, see [Setting up the Challenge-Handshake Authentication Protocol for the target](#).

Procedure

1. Enable CHAP authentication in the **iscsid.conf** file:

```
# vi /etc/iscsi/iscsid.conf  
  
node.session.auth.authmethod = CHAP
```

By default, the **node.session.auth.authmethod** is set to **None**

2. Add target **username** and **password** in the **iscsid.conf** file:

```
node.session.auth.username = redhat
node.session.auth.password = redhat_passwd
```

3. Start the **iscsid** daemon:

```
# systemctl start iscsid.service
```

Additional resources

- **iscsiadm(8)** man page

6.3. MONITORING AN ISCSI SESSION USING THE ISCSIADM UTILITY

This procedure describes how to monitor the iscsi session using the **iscsiadm** utility.

By default, an iSCSI service is **lazily** started and the service starts after running the **iscsiadm** command. If root is not on an iSCSI device or there are no nodes marked with **node.startup = automatic** then the iSCSI service will not start until an **iscsiadm** command is executed that requires **iscsid** or the **iscsi** kernel modules to be started.

Execute the **systemctl start iscsid.service** command as root to force the **iscsid** daemon to run and iSCSI kernel modules to load.

Procedure

1. Install the **iscsi-initiator-utils** on client machine:

```
# dnf install iscsi-initiator-utils
```

2. Find information about the running sessions:

```
# iscsiadm -m session -P 3
```

This command displays the session or device state, session ID (sid), some negotiated parameters, and the SCSI devices accessible through the session.

- For shorter output, for example, to display only the **sid-to-node** mapping, run:

```
# iscsiadm -m session -P 0
or
# iscsiadm -m session

tcp [2] 10.15.84.19:3260,2 iqn.1992-08.com.netapp:sn.33615311
tcp [3] 10.15.85.19:3260,3 iqn.1992-08.com.netapp:sn.33615311
```

These commands print the list of running sessions in the following format: **driver [sid] target_ip:port,target_portal_group_tag proper_target_name**.

Additional resources

- **/usr/share/doc/iscsi-initiator-utils-version/README** file

- **iscsiadm(8)** man page

6.4. DM MULTIPATH OVERRIDES OF THE DEVICE TIMEOUT

The **recovery_tmo sysfs** option controls the timeout for a particular iSCSI device. The following options globally override the **recovery_tmo** values:

- The **replacement_timeout** configuration option globally overrides the **recovery_tmo** value for all iSCSI devices.
- For all iSCSI devices that are managed by DM Multipath, the **fast_io_fail_tmo** option in DM Multipath globally overrides the **recovery_tmo** value.
The **fast_io_fail_tmo** option in DM Multipath also overrides the **fast_io_fail_tmo** option in Fibre Channel devices.

The DM Multipath **fast_io_fail_tmo** option takes precedence over **replacement_timeout**. Red Hat does not recommend using **replacement_timeout** to override **recovery_tmo** in devices managed by DM Multipath because DM Multipath always resets **recovery_tmo**, when the **multipathd** service reloads.

CHAPTER 7. USING FIBRE CHANNEL DEVICES

Red Hat Enterprise Linux 9 provides the following native Fibre Channel drivers:

- **lpfc**
- **qla2xxx**
- **zfcp**

7.1. RESIZING FIBRE CHANNEL LOGICAL UNITS

As a system administrator, you can resize Fibre Channel logical units.

Procedure

1. Determine which devices are paths for a **multipath** logical unit:

```
multipath -ll
```

2. Re-scan Fibre Channel logical units on a system that uses multipathing:

```
$ echo 1 > /sys/block/sdX/device/rescan
```

Additional resources

- **multipath(8)** man page

7.2. DETERMINING THE LINK LOSS BEHAVIOR OF DEVICE USING FIBRE CHANNEL

If a driver implements the Transport **dev_loss_tmo** callback, access attempts to a device through a link will be blocked when a transport problem is detected.

Procedure

- Determine the state of a remote port:

```
$ cat /sys/class/fc_remote_port/rport-host:bus:remote-port/port_state
```

This command returns any one of the following output:

- **Blocked** when the remote port along with devices accessed through it are blocked.
- **Online** if the remote port is operating normally
If the problem is not resolved within **dev_loss_tmo** seconds, the **rport** and devices will be unblocked. All I/O running on that device along with any new I/O sent to that device will fail.

When a link loss exceeds **dev_loss_tmo**, the **scsi_device** and **sd_N** devices are removed. Typically, the Fibre Channel class will leave the device as is, that is **/dev/sdx** will remain **/dev/sdx**. This is because the target binding is saved by the Fibre Channel driver and when the target port returns, the SCSI addresses are recreated faithfully. However, this cannot be guaranteed, the **sdx** device will be restored only if no additional change on in-storage box configuration of LUNs is made.

Additional resources

- **multipath.conf(5)** man page
- [Recommended tuning at scsi,multipath and at application layer while configuring Oracle RAC cluster](#) Knowledgebase article

7.3. FIBRE CHANNEL CONFIGURATION FILES

The following is the list of configuration files in the **/sys/class/** directory that provide the user-space API to Fibre Channel.

The items use the following variables:

H

Host number

B

Bus number

T

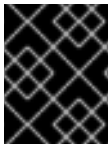
Target

L

Logical unit (LUNs)

R

Remote port number



IMPORTANT

If your system is using multipath software, Red Hat recommends that you consult your hardware vendor before changing any of the values described in this section.

Transport configuration in **/sys/class/fc_transport/targetH:B:T/**

port_id

24-bit port ID/address

node_name

64-bit node name

port_name

64-bit port name

Remote port configuration in **/sys/class/fc_remote_ports/rport-H:B-R/**

- **port_id**
- **node_name**
- **port_name**
- **dev_loss_tmo**

Controls when the scsi device gets removed from the system. After **dev_loss_tmo** triggers, the scsi device is removed. In the **multipath.conf** file, you can set **dev_loss_tmo** to **infinity**.

In Red Hat Enterprise Linux 9, if you do not set the **fast_io_fail_tmo** option, **dev_loss_tmo** is capped to **600** seconds. By default, **fast_io_fail_tmo** is set to **5** seconds in Red Hat Enterprise Linux 9 if the **multipathd** service is running; otherwise, it is set to **off**.

- **fast_io_fail_tmo**

Specifies the number of seconds to wait before it marks a link as "bad". Once a link is marked bad, existing running I/O or any new I/O on its corresponding path fails.

If I/O is in a blocked queue, it will not be failed until **dev_loss_tmo** expires and the queue is unblocked.

If **fast_io_fail_tmo** is set to any value except off, **dev_loss_tmo** is uncapped. If **fast_io_fail_tmo** is set to off, no I/O fails until the device is removed from the system. If **fast_io_fail_tmo** is set to a number, I/O fails immediately when the **fast_io_fail_tmo** timeout triggers.

Host configuration in **/sys/class/fc_host/hostH/**

- **port_id**
- **node_name**
- **port_name**
- **issue_lip**

Instructs the driver to rediscover remote ports.

7.4. DM MULTIPATH OVERRIDES OF THE DEVICE TIMEOUT

The **recovery_tmo sysfs** option controls the timeout for a particular iSCSI device. The following options globally override the **recovery_tmo** values:

- The **replacement_timeout** configuration option globally overrides the **recovery_tmo** value for all iSCSI devices.
- For all iSCSI devices that are managed by DM Multipath, the **fast_io_fail_tmo** option in DM Multipath globally overrides the **recovery_tmo** value.
The **fast_io_fail_tmo** option in DM Multipath also overrides the **fast_io_fail_tmo** option in Fibre Channel devices.

The DM Multipath **fast_io_fail_tmo** option takes precedence over **replacement_timeout**. Red Hat does not recommend using **replacement_timeout** to override **recovery_tmo** in devices managed by DM Multipath because DM Multipath always resets **recovery_tmo**, when the **multipathd** service reloads.

CHAPTER 8. NVME OVER FABRICS USING RDMA

In an NVMe over RDMA (NVMe/RDMA) setup, you configure an NVMe target and an NVMe initiator.

As a system administrator, complete the following tasks to deploy the NVMe/RDMA setup:

- [Setting up an NVMe/RDMA target using configs](#)
- [Setting up the NVMe/RDMA target using nvmetcli](#)
- [Configuring an NVMe/RDMA client](#)

8.1. OVERVIEW OF NVME OVER FABRIC DEVICES

Non-volatile Memory Express (NVMe) is an interface that allows host software utility to communicate with solid state drives.

Use the following types of fabric transport to configure NVMe over fabric devices:

NVMe over Remote Direct Memory Access (NVMe/RDMA)

For information on how to configure NVMe/RDMA, see [NVMe over fabrics using RDMA](#).

NVMe over Fibre Channel (FC-NVMe)

For information on how to configure FC-NVMe, see [NVMe over fabrics using FC](#).

When using Fibre Channel (FC) and Remote Direct Memory Access (RDMA), the solid-state drive does not have to be local to your system; it can be configured remotely through a FC or RDMA controller.

8.2. SETTING UP AN NVME/RDMA TARGET USING CONFIGFS

Use this procedure to configure an NVMe/RDMA target using **configs**.

Prerequisites

- Verify that you have a block device to assign to the **nvmet** subsystem.

Procedure

1. Create the **nvmet-rdma** subsystem:

```
# modprobe nvmet-rdma  
  
# mkdir /sys/kernel/config/nvmet/subsystems/testnqn  
  
# cd /sys/kernel/config/nvmet/subsystems/testnqn
```

Replace *testnqn* with the subsystem name.

2. Allow any host to connect to this target:

```
# echo 1 > attr_allow_any_host
```

3. Configure a namespace:


```
# mkdir namespaces/10

# cd namespaces/10
```

Replace *10* with the namespace number

4. Set a path to the NVMe device:

```
# echo -n /dev/nvme0n1 > device_path
```

5. Enable the namespace:

```
# echo 1 > enable
```

6. Create a directory with an NVMe port:

```
# mkdir /sys/kernel/config/nvmet/ports/1

# cd /sys/kernel/config/nvmet/ports/1
```

7. Display the IP address of *mlx5_ib0*:

```
# ip addr show mlx5_ib0

8: mlx5_ib0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 4092 qdisc mq state UP
group default qlen 256
    link/infiniband 00:00:06:2f:fe:80:00:00:00:00:00:00:e4:1d:2d:03:00:e7:0f:f6 brd
    00:ff:ff:ff:12:40:1b:ff:ff:00:00:00:00:00:00:00:ff:ff:ff
    inet 172.31.0.202/24 brd 172.31.0.255 scope global noprefixroute mlx5_ib0
        valid_lft forever preferred_lft forever
    inet6 fe80::e61d:2d03:e7:ff6/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
```

8. Set the transport address for the target:

```
# echo -n 172.31.0.202 > addr_traddr
```

9. Set RDMA as the transport type:

```
# echo rdma > addr_trtype

# echo 4420 > addr_trsvcid
```

10. Set the address family for the port:

```
# echo ipv4 > addr_adrfam
```

11. Create a soft link:

```
# ln -s /sys/kernel/config/nvmet/subsystems/testnqn
/sys/kernel/config/nvmet/ports/1/subsystems/testnqn
```

Verification

- Verify that the NVMe target is listening on the given port and ready for connection requests:

```
# dmesg | grep "enabling port"
[ 1091.413648] nvmet_rdma: enabling port 1 (172.31.0.202:4420)
```

Additional resources

- **nvme(1)** man page

8.3. SETTING UP THE NVME/RDMA TARGET USING NVMETCLI

Use the **nvmetcli** utility to edit, view, and start an NVMe target. The **nvmetcli** utility provides a command line and an interactive shell option. Use this procedure to configure the NVMe/RDMA target by **nvmetcli**.

Prerequisites

- Verify that you have a block device to assign to the **nvmet** subsystem.
- Execute the following **nvmetcli** operations as a root user.

Procedure

1. Install the **nvmetcli** package:

```
# dnf install nvmetcli
```

2. Download the **rdma.json** file:

```
# wget
http://git.infradead.org/users/hch/nvmetcli.git/blob_plain/0a6b088db2dc2e5de11e6f23f1e890e4b54fee64:/rdma.json
```

3. Edit the **rdma.json** file and change the **traddr** value to **172.31.0.202**.
4. Setup the target by loading the NVMe target configuration file:

```
# nvmetcli restore rdma.json
```



NOTE

If the NVMe target configuration file name is not specified, the **nvmetcli** uses the **/etc/nvmet/config.json** file.

Verification

- Verify that the NVMe target is listening on the given port and ready for connection requests:

```
# dmesg | tail -1
[ 4797.132647] nvmet_rdma: enabling port 2 (172.31.0.202:4420)
```

- Optional: Clear the current NVMe target:

```
# nvmetcli clear
```

Additional resources

- **nvmetcli** and **nvme(1)** man pages

8.4. CONFIGURING AN NVME/RDMA CLIENT

Use this procedure to configure an NVMe/RDMA client using the NVMe management command line interface (**nvme-cli**) tool.

Procedure

1. Install the **nvme-cli** tool:

```
# dnf install nvme-cli
```

2. Load the **nvme-rdma** module if it is not loaded:

```
# modprobe nvme-rdma
```

3. Discover available subsystems on the NVMe target:

```
# nvme discover -t rdma -a 172.31.0.202 -s 4420
```

```
Discovery Log Number of Records 1, Generation counter 2
```

```
=====Discovery Log Entry 0=====
```

```
trtype: rdma
```

```
adrfam: ipv4
```

```
subtype: nvme subsystem
```

```
treq: not specified, sq flow control disable supported
```

```
portid: 1
```

```
trsvcid: 4420
```

```
subnqn: testnqn
```

```
traddr: 172.31.0.202
```

```
rdma_prtype: not specified
```

```
rdma_qptype: connected
```

```
rdma_cms: rdma-cm
```

```
rdma_pkey: 0x0000
```

4. Connect to the discovered subsystems:

```
# nvme connect -t rdma -n testnqn -a 172.31.0.202 -s 4420
```

```
# lsblk
```

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

```
sda 8:0 0 465.8G 0 disk
```

```
└─sda1 8:1 0 1G 0 part /boot
```

```
└─sda2 8:2 0 464.8G 0 part
```

```
└─rhel_rdma--virt--03-root 253:0 0 50G 0 lvm /
```

```
└─rhel_rdma--virt--03-swap 253:1 0 4G 0 lvm [SWAP]
```

```
└─rhel_rdma--virt--03-home 253:2 0 410.8G 0 lvm /home
nvme0n1

# cat /sys/class/nvme/nvme0/transport
rdma
```

Replace *testnqn* with the NVMe subsystem name.

Replace *172.31.0.202* with the target IP address.

Replace *4420* with the port number.

Verification

- List the NVMe devices that are currently connected:

```
# nvme list
```

- Optional: Disconnect from the target:

```
# nvme disconnect -n testnqn
NQN:testnqn disconnected 1 controller(s)

# lsblk
NAME                                MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda                                8:0  0 465.8G 0 disk
├─sda1                            8:1  0    1G 0 part /boot
├─sda2                            8:2  0 464.8G 0 part
│   └─rhel_rdma--virt--03-root 253:0  0   50G 0 lvm /
│       └─rhel_rdma--virt--03-swap 253:1  0    4G 0 lvm [SWAP]
│           └─rhel_rdma--virt--03-home 253:2  0 410.8G 0 lvm /home
```

Additional resources

- nvme(1)** man page
- [Nvme-cli Github repository](#)

CHAPTER 9. NVME OVER FABRICS USING FC

The NVMe over Fibre Channel (FC-NVMe) transport is fully supported in initiator mode when used with certain Broadcom Emulex and Marvell Qlogic Fibre Channel adapters. As a system administrator, complete the tasks in the following sections to deploy the FC-NVMe setup:

- [Configuring the NVMe initiator for Broadcom adapters](#)
- [Configuring the NVMe initiator for QLogic adapters](#)

9.1. OVERVIEW OF NVME OVER FABRIC DEVICES

Non-volatile Memory Express (NVMe) is an interface that allows host software utility to communicate with solid state drives.

Use the following types of fabric transport to configure NVMe over fabric devices:

NVMe over Remote Direct Memory Access (NVMe/RDMA)

For information on how to configure NVMe/RDMA, see [NVMe over fabrics using RDMA](#).

NVMe over Fibre Channel (FC-NVMe)

For information on how to configure FC-NVMe, see [NVMe over fabrics using FC](#).

When using Fibre Channel (FC) and Remote Direct Memory Access (RDMA), the solid-state drive does not have to be local to your system; it can be configured remotely through a FC or RDMA controller.

9.2. CONFIGURING THE NVME INITIATOR FOR BROADCOM ADAPTERS

Use this procedure to configure the NVMe initiator for Broadcom adapters client using the NVMe management command line interface (**nvme-cli**) tool.

Procedure

1. Install the **nvme-cli** tool:

```
# dnf install nvme-cli
```

This creates the **hostnqn** file in the **/etc/nvme/** directory. The **hostnqn** file identifies the NVMe host.

To generate a new **hostnqn**, use the following command:

```
# nvme gen-hostnqn
```

2. Find the WWNN and WWPN identifiers of the local and remote ports and use the output to find the subsystem NQN:

```
# cat /sys/class/scsi_host/host*/nvme_info
```

```
NVME Initiator Enabled
XRI Dist lpfc0 Total 6144 IO 5894 ELS 250
NVME LPORT lpfc0 WWPN x10000090fae0b5f5 WWNN x20000090fae0b5f5 DID x010f00
```

```

ONLINE
NVME RPORT      WWPN x204700a098cbcac6 WWNN x204600a098cbcac6 DID x01050e
TARGET DISCSRV ONLINE

```

NVME Statistics

```

LS: Xmt 000000000e Cmpl 000000000e Abort 00000000
LS XMIT: Err 00000000 CMPL: xb 00000000 Err 00000000
Total FCP Cmpl 00000000000008ea Issue 00000000000008ec OutIO 0000000000000002
      abort 00000000 noxri 00000000 nondlp 00000000 qdepth 00000000 wqerr 00000000 err
00000000
FCP CMPL: xb 00000000 Err 00000000

```

```

# nvme discover --transport fc \
    --traddr nn-0x204600a098cbcac6:pn-0x204700a098cbcac6 \
    --host-traddr nn-0x20000090fae0b5f5:pn-0x10000090fae0b5f5

```

Discovery Log Number of Records 2, Generation counter 49530

=====Discovery Log Entry 0=====

```

trtype: fc
adrfam: fibre-channel
subtype: nvme subsystem
treq:   not specified
portid: 0
trsvcid: none
subnqn: nqn.1992-
08.com.netapp:sn.e18bfca87d5e11e98c0800a098cbcac6:subsystem.st14_nvme_ss_1_1
traddr: nn-0x204600a098cbcac6:pn-0x204700a098cbcac6

```

Replace `nn-0x204600a098cbcac6:pn-0x204700a098cbcac6` with the **traddr**.

Replace `nn-0x20000090fae0b5f5:pn-0x10000090fae0b5f5` with the **host-traddr**.

3. Connect to the NVMe target using the **nvme-cli**:

```

# nvme connect --transport fc \
    --traddr nn-0x204600a098cbcac6:pn-0x204700a098cbcac6 \
    --host-traddr nn-0x20000090fae0b5f5:pn-0x10000090fae0b5f5 \
    -n nqn.1992-
08.com.netapp:sn.e18bfca87d5e11e98c0800a098cbcac6:subsystem.st14_nvme_ss_1_1

```

Replace `nn-0x204600a098cbcac6:pn-0x204700a098cbcac6` with the **traddr**.

Replace `nn-0x20000090fae0b5f5:pn-0x10000090fae0b5f5` with the **host-traddr**.

Replace `nqn.1992-`

`08.com.netapp:sn.e18bfca87d5e11e98c0800a098cbcac6:subsystem.st14_nvme_ss_1_1` with the **subnqn**.

Verification

- List the NVMe devices that are currently connected:

```

# nvme list
Node      SN          Model          Namespace Usage
Format    FW Rev

```

```

-----
/dev/nvme0n1   80BgLFM7xMJbAAAAAAAC NetApp ONTAP Controller      1
107.37 GB / 107.37 GB   4 KiB + 0 B   FFFFFFFF

# lsblk |grep nvme
nvme0n1          259:0    0  100G  0 disk

```

Additional resources

- **nvme(1)** man page
- [Nvme-cli Github repository](#)

9.3. CONFIGURING THE NVME INITIATOR FOR QLOGIC ADAPTERS

Use this procedure to configure NVMe initiator for Qlogic adapters client using the NVMe management command line interface (**nvme-cli**) tool.

Procedure

1. Install the **nvme-cli** tool:

```
# dnf install nvme-cli
```

This creates the **hostnqn** file in the **/etc/nvme/** directory. The **hostnqn** file identifies the NVMe host.

To generate a new **hostnqn**, use the following command:

```
# nvme gen-hostnqn
```

2. Reload the **qla2xxx** module:

```
# rmmod qla2xxx
# modprobe qla2xxx
```

3. Find the WWNN and WWPN identifiers of the local and remote ports:

```
# dmesg |grep traddr

[ 6.139862] qla2xxx [0000:04:00.0]-ffff:0: register_localport: host-traddr=nn-
0x20000024ff19bb62:pn-0x21000024ff19bb62 on portID:10700
[ 6.241762] qla2xxx [0000:04:00.0]-2102:0: qla_nvme_register_remote: traddr=nn-
0x203b00a098cbcac6:pn-0x203d00a098cbcac6 PortID:01050d

```

Using these **host-traddr** and **traddr** values, find the subsystem NQN:

```
# nvme discover --transport fc \
    --traddr nn-0x203b00a098cbcac6:pn-0x203d00a098cbcac6 \
    --host-traddr nn-0x20000024ff19bb62:pn-0x21000024ff19bb62

```

Discovery Log Number of Records 2, Generation counter 49530

```
=====Discovery Log Entry 0=====
```

```
trtype: fc
adrfam: fibre-channel
subtype: nvme subsystem
treq: not specified
portid: 0
trsvcid: none
subnqn: nqn.1992-
08.com.netapp:sn.c9ecc9187b1111e98c0800a098cbcac6:subsystem.vs_nvme_multipath_1_subsystem_468
traddr: nn-0x203b00a098cbcac6:pn-0x203d00a098cbcac6
```

Replace `nn-0x203b00a098cbcac6:pn-0x203d00a098cbcac6` with the **traddr**.

Replace `nn-0x20000024ff19bb62:pn-0x21000024ff19bb62` with the **host-traddr**.

4. Connect to the NVMe target using the **nvme-cli** tool:

```
# nvme connect --transport fc \
    --traddr nn-0x203b00a098cbcac6:pn-0x203d00a098cbcac6 \
    --host-traddr nn-0x20000024ff19bb62:pn-0x21000024ff19bb62 \
    -n nqn.1992-
08.com.netapp:sn.c9ecc9187b1111e98c0800a098cbcac6:subsystem.vs_nvme_multipath_1_subsystem_468
```

Replace `nn-0x203b00a098cbcac6:pn-0x203d00a098cbcac6` with the **traddr**.

Replace `nn-0x20000024ff19bb62:pn-0x21000024ff19bb62` with the **host-traddr**.

Replace `nqn.1992-08.com.netapp:sn.c9ecc9187b1111e98c0800a098cbcac6:subsystem.vs_nvme_multipath_1_subsystem_468` with the **subnqn**.

Verification

- List the NVMe devices that are currently connected:

```
# nvme list
Node           SN           Model           Namespace Usage
Format         FW Rev
-----
/dev/nvme0n1   80BgLFM7xMJbAAAAAAAC NetApp ONTAP Controller 1
107.37 GB / 107.37 GB 4 KiB + 0 B FFFFFFFF

# lsblk |grep nvme
nvme0n1        259:0  0 100G 0 disk
```

Additional resources

- **nvme(1)** man page
- [Nvme-cli Github repository](#)

CHAPTER 10. ENABLING MULTIPATHING ON NVME DEVICES

You can multipath NVMe devices that are connected to your system over a fabric transport, such as Fibre Channel (FC). You can select between multiple multipathing solutions.

10.1. NATIVE NVME MULTIPATHING AND DM MULTIPATH

NVMe devices support a native multipathing functionality. When configuring multipathing on NVMe, you can select between the standard DM Multipath framework and the native NVMe multipathing.

Both DM Multipath and native NVMe multipathing support the Asymmetric Namespace Access (ANA) multipathing scheme of NVMe devices. ANA identifies optimized paths between the target and the initiator and improves performance.

When native NVMe multipathing is enabled, it applies globally to all NVMe devices. It can provide higher performance, but does not contain all of the functionality that DM Multipath provides. For example, native NVMe multipathing supports only the **failover** and **round-robin** path selection methods.

By default, NVMe multipathing is enabled in Red Hat Enterprise Linux 9 and is the recommended multipathing solution.

10.2. ENABLING NATIVE NVME MULTIPATHING

This procedure enables multipathing on connected NVMe devices using the native NVMe multipathing solution.

Prerequisites

- The NVMe devices are connected to your system.
For more information on connecting NVMe over fabric transports, see [Overview of NVMe over fabric devices](#).

Procedure

1. Check if native NVMe multipathing is enabled in the kernel:

```
# cat /sys/module/nvme_core/parameters/multipath
```

The command displays one of the following:

N

Native NVMe multipathing is disabled.

Y

Native NVMe multipathing is enabled.

2. If native NVMe multipathing is disabled, enable it using one of the following methods:

- Using a kernel option:
 - i. Add the **nvme_core.multipath=Y** option on the kernel command line:

```
# grubby --update-kernel=ALL --args="nvme_core.multipath=Y"
```

- ii. On the 64-bit IBM Z architecture, update the boot menu:

```
# zipl
```

- iii. Reboot the system.

- Using a kernel module configuration file:

- i. Create the **/etc/modprobe.d/nvme_core.conf** configuration file with the following content:

```
options nvme_core multipath=Y
```

- ii. Back up the **initramfs** file system:

```
# cp /boot/initramfs-$(uname -r).img \
  /boot/initramfs-$(uname -r).bak.$(date +%m-%d-%H%M%S).img
```

- iii. Rebuild the **initramfs** file system:

```
# dracut --force --verbose
```

- iv. Reboot the system.

3. Optional: On the running system, change the I/O policy on NVMe devices to distribute the I/O on all available paths:

```
# echo "round-robin" > /sys/class/nvme-subsystem/nvme-subsys0/iopolicy
```

4. Optional: Set the I/O policy persistently using **udev** rules. Create the **/etc/udev/rules.d/71-nvme-io-policy.rules** file with the following content:

```
ACTION=="add|change", SUBSYSTEM=="nvme-subsystem", ATTR{iopolicy}="round-robin"
```

Verification

1. Check that your system recognizes the NVMe devices:

```
# nvme list
```

Node Format	SN FW Rev	Model	Namespace Usage

/dev/nvme0n1	a34c4f3a0d6f5cec	Linux	1 250.06 GB /
250.06 GB	512 B + 0 B	4.18.0-2	
/dev/nvme0n2	a34c4f3a0d6f5cec	Linux	2 250.06 GB /
250.06 GB	512 B + 0 B	4.18.0-2	

2. List all connected NVMe subsystems:

```
# nvme list-subsys
```

```
nvme-subsys0 - NQN=testnqn
\
+- nvme0 fc traddr=nn-0x20000090fadd597a:pn-0x10000090fadd597a host_traddr=nn-
0x20000090fac7e1dd:pn-0x10000090fac7e1dd live
+- nvme1 fc traddr=nn-0x20000090fadd5979:pn-0x10000090fadd5979 host_traddr=nn-
0x20000090fac7e1dd:pn-0x10000090fac7e1dd live
+- nvme2 fc traddr=nn-0x20000090fadd5979:pn-0x10000090fadd5979 host_traddr=nn-
0x20000090fac7e1de:pn-0x10000090fac7e1de live
+- nvme3 fc traddr=nn-0x20000090fadd597a:pn-0x10000090fadd597a host_traddr=nn-
0x20000090fac7e1de:pn-0x10000090fac7e1de live
```

Check the active transport type. For example, **nvme0 fc** indicates that the device is connected over the Fibre Channel transport, and **nvme tcp** indicates that the device is connected over TCP.

3. If you edited the kernel options, check that native NVMe multipathing is enabled on the kernel command line:

```
# cat /proc/cmdline

BOOT_IMAGE=[...] nvme_core.multipath=Y
```

4. Check that DM Multipath reports the NVMe namespaces as, for example, **nvme0c0n1** through **nvme0c3n1**, and *not* as, for example, **nvme0n1** through **nvme3n1**:

```
# multipath -e -ll | grep -i nvme

uuid.8ef20f70-f7d3-4f67-8d84-1bb16b2bfe03 [nvme]:nvme0n1 NVMe,Linux,4.18.0-2
| ` 0:0:1  nvme0c0n1 0:0  n/a  optimized live
| ` 0:1:1  nvme0c1n1 0:0  n/a  optimized live
| ` 0:2:1  nvme0c2n1 0:0  n/a  optimized live
| ` 0:3:1  nvme0c3n1 0:0  n/a  optimized live

uuid.44c782b4-4e72-4d9e-bc39-c7be0a409f22 [nvme]:nvme0n2 NVMe,Linux,4.18.0-2
| ` 0:0:1  nvme0c0n1 0:0  n/a  optimized live
| ` 0:1:1  nvme0c1n1 0:0  n/a  optimized live
| ` 0:2:1  nvme0c2n1 0:0  n/a  optimized live
| ` 0:3:1  nvme0c3n1 0:0  n/a  optimized live
```

5. If you changed the I/O policy, check that **round-robin** is the active I/O policy on NVMe devices:

```
# cat /sys/class/nvme-subsystem/nvme-subsys0/iopolicy

round-robin
```

Additional resources

- [Configuring kernel command-line parameters](#)

10.3. ENABLING DM MULTIPATH ON NVME DEVICES

This procedure enables multipathing on connected NVMe devices using the DM Multipath solution.

Prerequisites

- The NVMe devices are connected to your system.
For more information on connecting NVMe over fabric transports, see [Overview of NVMe over fabric devices](#).

Procedure

1. Check that native NVMe multipathing is disabled:

```
# cat /sys/module/nvme_core/parameters/multipath
```

The command displays one of the following:

N

Native NVMe multipathing is disabled.

Y

Native NVMe multipathing is enabled.

2. If native NVMe multipathing is enabled, disable it:
 - a. Remove the **nvme_core.multipath=Y** option from the kernel command line:

```
# grubby --update-kernel=ALL --remove-args="nvme_core.multipath=Y"
```

- b. On the 64-bit IBM Z architecture, update the boot menu:

```
# zipl
```

- c. Remove the **options nvme_core multipath=Y** line from the **/etc/modprobe.d/nvme_core.conf** file, if it is present.
- d. Reboot the system.

3. Make sure that DM Multipath is enabled:

```
# systemctl enable --now multipathd.service
```

4. Distribute I/O on all available paths. Add the following content in the **/etc/multipath.conf** file:

```
device {
    vendor "NVME"
    product ".*"
    path_grouping_policy  group_by_prio
}
```



NOTE

The **/sys/class/nvme-subsystem/nvme-subsys0/iopolicy** configuration file has no effect on the I/O distribution when DM Multipath manages the NVMe devices.

5. Reload the **multipathd** service to apply the configuration changes:

```
# multipath -r
```

- Back up the **initramfs** file system:

```
# cp /boot/initramfs-$(uname -r).img \
  /boot/initramfs-$(uname -r).bak.$(date +%m-%d-%H%M%S).img
```

- Rebuild the **initramfs** file system:

```
# dracut --force --verbose
```

Verification

- Check that your system recognizes the NVMe devices:

```
# nvme list
```

Node Format	SN FW Rev	Model	Namespace Usage
/dev/nvme0n1	a34c4f3a0d6f5cec	Linux	1 250.06 GB /
250.06 GB	512 B + 0 B	4.18.0-2	
/dev/nvme0n2	a34c4f3a0d6f5cec	Linux	2 250.06 GB /
250.06 GB	512 B + 0 B	4.18.0-2	
/dev/nvme1n1	a34c4f3a0d6f5cec	Linux	1 250.06 GB /
250.06 GB	512 B + 0 B	4.18.0-2	
/dev/nvme1n2	a34c4f3a0d6f5cec	Linux	2 250.06 GB /
250.06 GB	512 B + 0 B	4.18.0-2	
/dev/nvme2n1	a34c4f3a0d6f5cec	Linux	1 250.06 GB /
250.06 GB	512 B + 0 B	4.18.0-2	
/dev/nvme2n2	a34c4f3a0d6f5cec	Linux	2 250.06 GB /
250.06 GB	512 B + 0 B	4.18.0-2	
/dev/nvme3n1	a34c4f3a0d6f5cec	Linux	1 250.06 GB /
250.06 GB	512 B + 0 B	4.18.0-2	
/dev/nvme3n2	a34c4f3a0d6f5cec	Linux	2 250.06 GB /
250.06 GB	512 B + 0 B	4.18.0-2	

- List all connected NVMe subsystems. Check that the command reports them as, for example, **nvme0n1** through **nvme3n2**, and *not* as, for example, **nvme0c0n1** through **nvme0c3n1**:

```
# nvme list-subsys
```

```
nvme-subsys0 - NQN=testnqn
\
+- nvme0 fc traddr=nn-0x20000090fadd5979:pn-0x10000090fadd5979 host_traddr=nn-
0x20000090fac7e1dd:pn-0x10000090fac7e1dd live
+- nvme1 fc traddr=nn-0x20000090fadd597a:pn-0x10000090fadd597a host_traddr=nn-
0x20000090fac7e1dd:pn-0x10000090fac7e1dd live
+- nvme2 fc traddr=nn-0x20000090fadd5979:pn-0x10000090fadd5979 host_traddr=nn-
0x20000090fac7e1de:pn-0x10000090fac7e1de live
+- nvme3 fc traddr=nn-0x20000090fadd597a:pn-0x10000090fadd597a host_traddr=nn-
0x20000090fac7e1de:pn-0x10000090fac7e1de live
```

```
# multipath -ll
```

```
mpathae (uuid.8ef20f70-f7d3-4f67-8d84-1bb16b2bfe03) dm-36 NVME,Linux
size=233G features='1 queue_if_no_path' hwhandler='0' wp=rw
`-+- policy='service-time 0' prio=50 status=active
   |- 0:1:1:1 nvme0n1 259:0 active ready running
   |- 1:2:1:1 nvme1n1 259:2 active ready running
   |- 2:3:1:1 nvme2n1 259:4 active ready running
   `-- 3:4:1:1 nvme3n1 259:6 active ready running

mpathaf (uuid.44c782b4-4e72-4d9e-bc39-c7be0a409f22) dm-39 NVME,Linux
size=233G features='1 queue_if_no_path' hwhandler='0' wp=rw
`-+- policy='service-time 0' prio=50 status=active
   |- 0:1:2:2 nvme0n2 259:1 active ready running
   |- 1:2:2:2 nvme1n2 259:3 active ready running
   |- 2:3:2:2 nvme2n2 259:5 active ready running
   `-- 3:4:2:2 nvme3n2 259:7 active ready running
```

Additional resources

- [Configuring kernel command-line parameters](#)
- [Configuring DM Multipath.](#)

CHAPTER 11. GETTING STARTED WITH SWAP

This section describes swap space, and how to add and remove it.

11.1. OVERVIEW OF SWAP SPACE

Swap space in Linux is used when the amount of physical memory (RAM) is full. If the system needs more memory resources and the RAM is full, inactive pages in memory are moved to the swap space. While swap space can help machines with a small amount of RAM, it should not be considered a replacement for more RAM.

Swap space is located on hard drives, which have a slower access time than physical memory. Swap space can be a dedicated swap partition (recommended), a swap file, or a combination of swap partitions and swap files.

In years past, the recommended amount of swap space increased linearly with the amount of RAM in the system. However, modern systems often include hundreds of gigabytes of RAM. As a consequence, recommended swap space is considered a function of system memory workload, not system memory.

Adding swap space

The following are the different ways to add a swap space:

- [Extending swap on an LVM2 logical volume](#)
- [Creating an LVM2 logical volume for swap](#)
- [Creating a swap file](#)

For example, you may upgrade the amount of RAM in your system from 1 GB to 2 GB, but there is only 2 GB of swap space. It might be advantageous to increase the amount of swap space to 4 GB if you perform memory-intensive operations or run applications that require a large amount of memory.

Removing swap space

The following are the different ways to remove a swap space:

- [Reducing swap on an LVM2 logical volume](#)
- [Removing an LVM2 logical volume for swap](#)
- [Removing a swap file](#)

For example, you have downgraded the amount of RAM in your system from 1 GB to 512 MB, but there is 2 GB of swap space still assigned. It might be advantageous to reduce the amount of swap space to 1 GB, since the larger 2 GB could be wasting disk space.

11.2. RECOMMENDED SYSTEM SWAP SPACE

This section describes the recommended size of a swap partition depending on the amount of RAM in your system and whether you want sufficient memory for your system to hibernate. The recommended swap partition size is established automatically during installation. To allow for hibernation, however, you need to edit the swap space in the custom partitioning stage.

The following recommendation are especially important on systems with low memory such as 1 GB and less. Failure to allocate sufficient swap space on these systems can cause issues such as instability or even render the installed system unbootable.

Table 11.1. Recommended swap space

Amount of RAM in the system	Recommended swap space	Recommended swap space if allowing for hibernation
≤ 2 GB	2 times the amount of RAM	3 times the amount of RAM
> 2 GB – 8 GB	Equal to the amount of RAM	2 times the amount of RAM
> 8 GB – 64 GB	At least 4 GB	1.5 times the amount of RAM
> 64 GB	At least 4 GB	Hibernation not recommended

At the border between each range listed in this table, for example a system with 2 GB, 8 GB, or 64 GB of system RAM, discretion can be exercised with regard to chosen swap space and hibernation support. If your system resources allow for it, increasing the swap space may lead to better performance.

Note that distributing swap space over multiple storage devices also improves swap space performance, particularly on systems with fast drives, controllers, and interfaces.



IMPORTANT

File systems and LVM2 volumes assigned as swap space *should not* be in use when being modified. Any attempts to modify swap fail if a system process or the kernel is using swap space. Use the **free** and **cat /proc/swaps** commands to verify how much and where swap is in use.

Resizing swap space requires temporarily removing the swap space from the system. This can be problematic if running applications rely on the additional swap space and might run into low-memory situations. Preferably, perform swap resizing from rescue mode, see [Debug boot options](#) in the *Performing an advanced RHEL 9 installation*. When prompted to mount the file system, select **Skip**.

11.3. EXTENDING SWAP ON AN LVM2 LOGICAL VOLUME

This procedure describes how to extend swap space on an existing LVM2 logical volume. Assuming `/dev/VolGroup00/LogVol01` is the volume you want to extend by 2 GB.

Prerequisites

- You have sufficient disk space.

Procedure

1. Disable swapping for the associated logical volume:

```
# swapoff -v /dev/VolGroup00/LogVol01
```

2. Resize the LVM2 logical volume by 2 GB:

```
# lvresize /dev/VolGroup00/LogVol01 -L +2G
```


3. Format the new swap space:

```
# mkswap /dev/VolGroup00/LogVol01
```

4. Enable the extended logical volume:

```
# swapon -v /dev/VolGroup00/LogVol01
```

Verification

- To test if the swap logical volume was successfully extended and activated, inspect active swap space by using the following command:

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

11.4. CREATING AN LVM2 LOGICAL VOLUME FOR SWAP

This procedure describes how to create an LVM2 logical volume for swap. Assuming `/dev/VolGroup00/LogVol02` is the swap volume you want to add.

Prerequisites

- You have sufficient disk space.

Procedure

1. Create the LVM2 logical volume of size 2 GB:

```
# lvcreate VolGroup00 -n LogVol02 -L 2G
```

2. Format the new swap space:

```
# mkswap /dev/VolGroup00/LogVol02
```

3. Add the following entry to the `/etc/fstab` file:

```
/dev/VolGroup00/LogVol02 swap swap defaults 0 0
```

4. Regenerate mount units so that your system registers the new configuration:

```
# systemctl daemon-reload
```

5. Activate swap on the logical volume:

```
# swapon -v /dev/VolGroup00/LogVol02
```

Verification

- To test if the swap logical volume was successfully created and activated, inspect active swap space by using the following command:

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

11.5. CREATING A SWAP FILE

This procedure describes how to create a swap file.

Prerequisites

- You have sufficient disk space.

Procedure

1. Determine the size of the new swap file in megabytes and multiply by 1024 to determine the number of blocks. For example, the block size of a 64 MB swap file is 65536.

2. Create an empty file:

```
# dd if=/dev/zero of=/swapfile bs=1024 count=65536
```

Replace 65536 with the value equal to the desired block size.

3. Set up the swap file with the command:

```
# mkswap /swapfile
```

4. Change the security of the swap file so it is not world readable.

```
# chmod 0600 /swapfile
```

5. Edit the **/etc/fstab** file with the following entries to enable the swap file at boot time:

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

The next time the system boots, it activates the new swap file.

6. Regenerate mount units so that your system registers the new **/etc/fstab** configuration:

```
# systemctl daemon-reload
```

7. Activate the swap file immediately:

```
# swapon /swapfile
```

Verification

- To test if the new swap file was successfully created and activated, inspect active swap space by using the following command:

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

11.6. REDUCING SWAP ON AN LVM2 LOGICAL VOLUME

This procedure describes how to reduce swap on an LVM2 logical volume. Assuming `/dev/VolGroup00/LogVol01` is the volume you want to reduce.

Procedure

1. Disable swapping for the associated logical volume:

```
# swapoff -v /dev/VolGroup00/LogVol01
```

2. Reduce the LVM2 logical volume by 512 MB:

```
# lvreduce /dev/VolGroup00/LogVol01 -L -512M
```

3. Format the new swap space:

```
# mkswap /dev/VolGroup00/LogVol01
```

4. Activate swap on the logical volume:

```
# swapon -v /dev/VolGroup00/LogVol01
```

Verification

- To test if the swap logical volume was successfully reduced, inspect active swap space by using the following command:

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

11.7. REMOVING AN LVM2 LOGICAL VOLUME FOR SWAP

This procedure describes how to remove an LVM2 logical volume for swap. Assuming `/dev/VolGroup00/LogVol02` is the swap volume you want to remove.

Procedure

1. Disable swapping for the associated logical volume:

```
# swapoff -v /dev/VolGroup00/LogVol02
```

2. Remove the LVM2 logical volume:

```
# lvremove /dev/VolGroup00/LogVol02
```

3. Remove the following associated entry from the `/etc/fstab` file:

```
/dev/VolGroup00/LogVol02 swap swap defaults 0 0
```

4. Regenerate mount units so that your system registers the new configuration:

```
# systemctl daemon-reload
```

Verification

- To test if the logical volume was successfully removed, inspect active swap space by using the following command:

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

11.8. REMOVING A SWAP FILE

This procedure describes how to remove a swap file.

Procedure

1. At a shell prompt, execute the following command to disable the swap file, where **/swapfile** is the swap file:

```
# swapoff -v /swapfile
```

2. Remove its entry from the **/etc/fstab** file accordingly.
3. Regenerate mount units so that your system registers the new configuration:

```
# systemctl daemon-reload
```

4. Remove the actual file:

```
# rm /swapfile
```

CHAPTER 12. CONFIGURING FIBRE CHANNEL OVER ETHERNET

Based on the IEEE T11 FC-BB-5 standard, Fibre Channel over Ethernet (FCoE) is a protocol to transmit Fibre Channel frames over Ethernet networks. Typically, data centers have a dedicated LAN and Storage Area Network (SAN) that are separated from each other with their own specific configuration. FCoE combines these networks into a single and converged network structure. Benefits of FCoE are, for example, lower hardware and energy costs.

12.1. USING HARDWARE FCOE HBAS IN RHEL

In RHEL you can use hardware Fibre Channel over Ethernet (FCoE) Host Bus Adapter (HBA), which is supported by the following drivers:

- **qedf**
- **bnx2fc**
- **fnic**

If you use such a HBA, you configure the FCoE settings in the setup of the HBA. For more information, see the documentation of the adapter.

After you configure the HBA, the exported Logical Unit Numbers (LUN) from the Storage Area Network (SAN) are automatically available to RHEL as **/dev/sd*** devices. You can use these devices similar to local storage devices.

12.2. SETTING UP A SOFTWARE FCOE DEVICE

Use the software FCoE device to access Logical Unit Numbers (LUN) over FCoE, which uses using an Ethernet adapter that partially supports FCoE offload.



IMPORTANT

RHEL does not support software FCoE devices that require the **fcoe.ko** kernel module.

After you complete this procedure, the exported LUNs from the Storage Area Network (SAN) are automatically available to RHEL as **/dev/sd*** devices. You can use these devices in a similar way to local storage devices.

Prerequisites

- You have configured the network switch to support VLAN.
- The SAN uses a VLAN to separate the storage traffic from normal Ethernet traffic.
- You have configured the HBA of the server in its BIOS.
- The HBA is connected to the network and the link is up. For more information, see the documentation of your HBA.

Procedure

1. Install the **fcoe-utils** package:

```
# dnf install fcoe-utils
```

2. Copy the **/etc/fcoe/cfg-ethx** template file to **/etc/fcoe/cfg-interface_name**. For example, if you want to configure the **enp1s0** interface to use FCoE, enter the following command:

```
# cp /etc/fcoe/cfg-ethx /etc/fcoe/cfg-enp1s0
```

3. Enable and start the **fcoe** service:

```
# systemctl enable --now fcoe
```

4. Discover the FCoE VLAN on interface **enp1s0**, create a network device for the discovered VLAN, and start the initiator:

```
# fipvlan -s -c enp1s0
Created VLAN device enp1s0.200
Starting FCoE on interface enp1s0.200
Fibre Channel Forwarders Discovered
interface      | VLAN | FCF MAC
-----
enp1s0         | 200  | 00:53:00:a7:e7:1b
```

5. Optional: Display details about the discovered targets, the LUNs, and the devices associated with the LUNs:

```
# fcoeadm -t
Interface:      enp1s0.200
Roles:          FCP Target
Node Name:      0x500a0980824acd15
Port Name:      0x500a0982824acd15
Target ID:      0
MaxFrameSize:   2048 bytes
OS Device Name:  rport-11:0-1
FC-ID (Port ID): 0xba00a0
State:          Online

LUN ID Device Name Capacity Block Size Description
-----
0 sdb    28.38 GiB   512    NETAPP LUN (rev 820a)
...
```

This example shows that LUN 0 from the SAN has been attached to the host as the **/dev/sdb** device.

Verification

- Display information about all active FCoE interfaces:

```
# fcoeadm -i
Description:    BCM57840 NetXtreme II 10 Gigabit Ethernet
Revision:       11
Manufacturer:   Broadcom Inc. and subsidiaries
```

Serial Number: 000AG703A9B7

Driver: bnx2x Unknown

Number of Ports: 1

Symbolic Name: bnx2fc (QLogic BCM57840) v2.12.13 over enp1s0.200

OS Device Name: host11

Node Name: 0x2000000af70ae935

Port Name: 0x2001000af70ae935

Fabric Name: 0x20c8002a6aa7e701

Speed: 10 Gbit

Supported Speed: 1 Gbit, 10 Gbit

MaxFrameSize: 2048 bytes

FC-ID (Port ID): 0xba02c0

State: Online

Additional resources

- **fcoeadm(8)** man page
- **/usr/share/doc/fcoe-utils/README**
- [Using Fibre Channel devices](#)

CHAPTER 13. MANAGING TAPE DEVICES

A tape device is a magnetic tape where data is stored and accessed sequentially. Data is written to this tape device with the help of a tape drive. There is no need to create a file system in order to store data on a tape device. Tape drives can be connected to a host computer with various interfaces like, SCSI, FC, USB, SATA, and other interfaces.

13.1. TYPES OF TAPE DEVICES

The following is a list of the different types of tape devices:

- **/dev/st0** is a rewinding tape device.
- **/dev/nst0** is a non-rewinding tape device. Use non-rewinding devices for daily backups.

There are several advantages to using tape devices. They are cost efficient and stable. Tape devices are also resilient against data corruption and are suitable for data retention.

13.2. INSTALLING TAPE DRIVE MANAGEMENT TOOL

Use the **mt** command to wind the data back and forth. The **mt** utility controls magnetic tape drive operations and the **st** utility is used for SCSI tape driver. This procedure describes how to install the **mt-st** package for tape drive operations.

Procedure

- Install the **mt-st** package:

```
# dnf install mt-st
```

Additional resources

- **mt(1)** and **st(4)** man pages

13.3. WRITING TO REWINDING TAPE DEVICES

A rewind tape device rewinds the tape after every operation. To back up data, you can use the **tar** command. By default, in tape devices the **block size** is 10KB (**bs=10k**). You can set the **TAPE** environment variable using the **export TAPE=/dev/st0** attribute. Use the **-f** device option instead, to specify the tape device file. This option is useful when you use more than one tape device.

Prerequisites

1. You have installed the **mt-st** package. For more information, see [Installing tape drive management tool](#).
2. Load the tape drive:

```
# mt -f /dev/st0 load
```

Procedure

1. Check the tape head:

```
# mt -f /dev/st0 status
```

SCSI 2 tape drive:

File number=-1, block number=-1, partition=0.

Tape block size 0 bytes. Density code 0x0 (default).

Soft error count since last status=0

General status bits on (50000):

DR_OPEN IM_REP_EN

Here:

- the current **file number** is -1.
- the **block number** defines the tape head. By default, it is set to -1.
- the **block size** 0 indicates that the tape device does not have a fixed block size.
- the **Soft error count** indicates the number of encountered errors after executing the `mt status` command.
- the **General status bits** explains the stats of the tape device.
- **DR_OPEN** indicates that the door is open and the tape device is empty. **IM_REP_EN** is the immediate report mode.

2. If the tape device is not empty, overwrite it:

```
# tar -czf /dev/st0 _/source/directory
```

This command overwrites the data on a tape device with the content of **/source/directory**.

3. Back up the **/source/directory** to the tape device:

```
# tar -czf /dev/st0 _/source/directory
tar: Removing leading `/' from member names
/source/directory
/source/directory/man_db.conf
/source/directory/DIR_COLORS
/source/directory/rsyslog.conf
[...]
```

4. View the status of the tape device:

```
# mt -f /dev/st0 status
```

Verification steps

- View the list of all files on the tape device:

```
# tar -tzf /dev/st0
/source/directory/
/source/directory/man_db.conf
```

```
/source/directory/DIR_COLORS
/source/directory/rsyslog.conf
[...]
```

Additional resources

- **mt(1)**, **st(4)**, and **tar(1)** man pages
- [Tape drive media detected as write protected](#) Red Hat Knowledgebase article
- [How to check if tape drives are detected in the system](#) Red Hat Knowledgebase article

13.4. WRITING TO NON-REWINDING TAPE DEVICES

A non-rewinding tape device leaves the tape in its current status, after completing the execution of a certain command. For example, after a backup, you could append more data to a non-rewinding tape device. You can also use it to avoid any unexpected rewinds.

Prerequisites

1. You have installed the **mt-st** package. For more information, see [Installing tape drive management tool](#).
2. Load the tape drive:

```
# mt -f /dev/nst0 load
```

Procedure

1. Check the tape head of the non-rewinding tape device **/dev/nst0**:

```
# mt -f /dev/nst0 status
```

2. Specify the pointer at the head or at the end of the tape:

```
# mt -f /dev/nst0 rewind
```

3. Append the data on the tape device:

```
# mt -f /dev/nst0 eod
# tar -czf /dev/nst0 /source/directory/
```

4. Back up the **/source/directory/** to the tape device:

```
# tar -czf /dev/nst0 /source/directory/
tar: Removing leading `/' from member names
/source/directory/
/source/directory/man_db.conf
/source/directory/DIR_COLORS
/source/directory/rsyslog.conf
[...]
```

5. View the status of the tape device:

```
# mt -f /dev/nst0 status
```

Verification steps

- View the list of all files on the tape device:

```
# tar -tzf /dev/nst0
/source/directory/
/source/directory/man_db.conf
/source/directory/DIR_COLORS
/source/directory/rsyslog.conf
[...]
```

Additional resources

- **mt(1)**, **st(4)**, and **tar(1)** man pages
- [Tape drive media detected as write protected](#) Red Hat Knowledgebase article
- [How to check if tape drives are detected in the system](#) Red Hat Knowledgebase article

13.5. SWITCHING TAPE HEAD IN TAPE DEVICES

Use the following procedure to switch the tape head in the tape device.

Prerequisites

1. You have installed the **mt-st** package. For more information, see [Installing tape drive management tool](#).
2. Data is written to the tape device. For more information, see [Writing to rewinding tape devices](#) or [Writing to non-rewinding tape devices](#).

Procedure

- To view the current position of the tape pointer:

```
# mt -f /dev/nst0 tell
```

- To switch the tape head, while appending the data to the tape devices:

```
# mt -f /dev/nst0 eod
```

- To go to the previous record:

```
# mt -f /dev/nst0 bsfm 1
```

- To go to the forward record:

```
# mt -f /dev/nst0 fsf 1
```

Additional resources

- **mt(1)** man page

13.6. RESTORING DATA FROM TAPE DEVICES

To restore data from a tape device, use the **tar** command.

Prerequisites

1. You have installed the **mt-st** package. For more information, see [Installing tape drive management tool](#).
2. Data is written to the tape device. For more information, see [Writing to rewinding tape devices](#) or [Writing to non-rewinding tape devices](#).

Procedure

- For rewinding tape devices **/dev/st0**:

- Restore the **/source/directory/**:

```
# tar -xzf /dev/st0 /source/directory/
```

- For non-rewinding tape devices **/dev/nst0**:

- Rewind the tape device:

```
# mt -f /dev/nst0 rewind
```

- Restore the **etc** directory:

```
# tar -xzf /dev/nst0 /source/directory/
```

Additional resources

- **mt(1)** and **tar(1)** man pages

13.7. ERASING DATA FROM TAPE DEVICES

To erase data from a tape device, use the **erase** option.

Prerequisites

1. You have installed the **mt-st** package. For more information, see [Installing tape drive management tool](#).
2. Data is written to the tape device. For more information, see [Writing to rewinding tape devices](#) or [Writing to non-rewinding tape devices](#).

Procedure

1. Erase data from the tape device:

```
# mt -f /dev/st0 erase
```

2. Unload the tape device:

```
mt -f /dev/st0 offline
```

Additional resources

- **mt(1)** man page

13.8. TAPE COMMANDS

The following are the common **mt** commands:

Table 13.1. **mt** commands

Command	Description
mt -f /dev/st0 status	Displays the status of the tape device.
mt -f /dev/st0 erase	Erases the entire tape.
mt -f /dev/nst0 rewind	Rewinds the tape device.
mt -f /dev/nst0 fsf <i>n</i>	Switches the tape head to the forward record. Here, <i>n</i> is an optional file count. If a file count is specified, tape head skips <i>n</i> records.
mt -f /dev/nst0 bsfm <i>n</i>	Switches the tape head to the previous record.
mt -f /dev/nst0 eod	Switches the tape head to the end of the data.