Managing storage devices

Deploying and configuring single-node storage in Red Hat Enterprise Linux 9
Deploying and configuring single-node storage in Red Hat Enterprise Linux 9
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. CONFIGURING AN ISCSI TARGET .......................................... 11
  2.1. INSTALLING TARGETCLI ................................................ 11
  2.2. CREATING AN ISCSI TARGET ........................................... 12
  2.3. ISCSI BACKSTORE ...................................................... 13
  2.4. CREATING A FILEIO STORAGE OBJECT .............................. 13
  2.5. CREATING A BLOCK STORAGE OBJECT .............................. 14
  2.6. CREATING A PSCSI STORAGE OBJECT ............................... 15
  2.7. CREATING A MEMORY COPY RAM DISK STORAGE OBJECT ............ 15
  2.8. CREATING AN ISCSI PORTAL .......................................... 16
  2.9. CREATING AN ISCSI LUN .............................................. 17
  2.10. CREATING A READ-ONLY ISCSI LUN ................................. 18
  2.11. CREATING AN ISCSI ACL ............................................ 19
  2.12. SETTING UP THE CHALLENGE-HANDSHAKE AUTHENTICATION PROTOCOL FOR THE TARGET ........................... 21
  2.13. REMOVING AN ISCSI OBJECT USING TARGETCLI TOOL .............. 21

CHAPTER 3. CONFIGURING AN ISCSI INITIATOR ..................................... 23
  3.1. CREATING AN ISCSI INITIATOR ....................................... 23
  3.2. SETTING UP THE CHALLENGE-HANDSHAKE AUTHENTICATION PROTOCOL FOR THE INITIATOR ................................. 24
  3.3. MONITORING AN ISCSI SESSION USING THE ISCSIADM UTILITY ... 25
  3.4. DM MULTIPATH OVERRIDES OF THE DEVICE TIMEOUT .................. 26

CHAPTER 4. USING FIBRE CHANNEL DEVICES ....................................... 27
  4.1. RESIZING FIBRE CHANNEL LOGICAL UNITS ............................ 27
  4.2. DETERMINING THE LINK LOSS BEHAVIOR OF DEVICE USING FIBRE CHANNEL ......................................................... 27
  4.3. FIBRE CHANNEL CONFIGURATION FILES .............................. 28
  4.4. DM MULTIPATH OVERRIDES OF THE DEVICE TIMEOUT .................. 29

CHAPTER 5. NVME OVER FABRICS USING RDMA .................................... 30
  5.1. OVERVIEW OF NVME OVER FABRIC DEVICES .......................... 30
  5.2. SETTING UP AN NVME/RDMA TARGET USING CONFIGFS ............ 30
  5.3. SETTING UP THE NVME/RDMA TARGET USING NVMETCLI ............ 32
  5.4. CONFIGURING AN NVME/RDMA CLIENT ................................ 33

CHAPTER 6. NVME OVER FABRICS USING FC ....................................... 35
  6.1. OVERVIEW OF NVME OVER FABRIC DEVICES .......................... 35
  6.2. CONFIGURING THE NVME INITIATOR FOR BROADCOM ADAPTERS ... 35
  6.3. CONFIGURING THE NVME INITIATOR FOR QLOGIC ADAPTERS .......... 37

CHAPTER 7. ENABLING MULTIPATHING ON NVME DEVICES .......................... 39
  7.1. NATIVE NVME MULTIPATHING AND DM MULTIPATH .................... 39
  7.2. ENABLING NATIVE NVME MULTIPATHING ................................ 39
  7.3. ENABLING DM MULTIPATH ON NVME DEVICES .......................... 41
CHAPTER 8. GETTING STARTED WITH SWAP .................................................. 45
  8.1. OVERVIEW OF SWAP SPACE .......................................................... 45
  8.2. RECOMMENDED SYSTEM SWAP SPACE ......................................... 45
  8.3. EXTENDING SWAP ON AN LVM2 LOGICAL VOLUME ......................... 46
  8.4. CREATING AN LVM2 LOGICAL VOLUME FOR SWAP ......................... 47
  8.5. CREATING A SWAP FILE ............................................................... 48
  8.6. REDUCING SWAP ON AN LVM2 LOGICAL VOLUME .............................. 49
  8.7. REMOVING AN LVM2 LOGICAL VOLUME FOR SWAP ......................... 49
  8.8. REMOVING A SWAP FILE ............................................................... 50

CHAPTER 9. CONFIGURING FIBRE CHANNEL OVER ETHERNET ....................... 51
  9.1. USING HARDWARE FCOE HBAS IN RHEL ........................................ 51
  9.2. SETTING UP A SOFTWARE FCOE DEVICE ......................................... 51

CHAPTER 10. MANAGING TAPE DEVICES ...................................................... 54
  10.1. TYPES OF TAPE DEVICES ............................................................. 54
  10.2. INSTALLING TAPE DRIVE MANAGEMENT TOOL ................................... 54
  10.3. WRITING TO REWINDING TAPE DEVICES ........................................ 54
  10.4. WRITING TO NON-REWINDING TAPE DEVICES .................................. 56
  10.5. SWITCHING TAPE HEAD IN TAPE DEVICES ....................................... 57
  10.6. RESTORING DATA FROM TAPE DEVICES ......................................... 58
  10.7. ERASING DATA FROM TAPE DEVICES ............................................. 58
  10.8. TAPE COMMANDS ........................................................................ 59
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.
We appreciate your input on our documentation. Please let us know how we could make it better.

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

- For submitting feedback via Bugzilla, create a new ticket:
  1. Go to the Bugzilla website.
  2. As the Component, use Documentation.
  3. Fill in the Description field with your suggestion for improvement. Include a link to the relevant part(s) of documentation.
  4. Click Submit Bug.
CHAPTER 1. 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

1.1. LOCAL STORAGE OVERVIEW

RHEL Host

External SAN/NAS Storage

NIC/HBA cards/RAID cards

Device Drivers

Raw Device Partitioning

Volume Management Functions

File Systems

Applications

Storage Target Modes (NFS/CIFS/ISCSI)

External Systems

User

Persistent Memory Direct Access Path

Direct Attached/Internal SSD/Internal HDD

Persistent Memory (NVDIMM)

System-Resident Physical Storage Media
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. 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.

2.1. INSTALLING TARGETCLI

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

**Procedure**

1. Install the `targetcli` tool:
   
   ```bash
   # dnf install targetcli
   ```

2. Start the target service:
   
   ```bash
   # systemctl start target
   ```

3. Configure target to start at boot time:
   
   ```bash
   # systemctl enable target
   ```

4. Open port **3260** in the firewall and reload the firewall configuration:
   
   ```bash
   # firewall-cmd --permanent --add-port=3260/tcp
   Success
   # firewall-cmd --reload
   Success
   ```

**Verification**

- View the `targetcli` layout:

  ```bash
  # 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]
  ```
2.2. 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]
   ```
2.3. 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

2.4. 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

2.5. 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-I/O (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
2.6. 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

2.7. 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

2.8. 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:
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................................. [enabled, 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`

### 2.9. 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:

   ```bash
   /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:

   ```bash
   /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**

**2.10. 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:

   ```
   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

2.11. 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
2.12. 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

2.13. 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 3. 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.

3.1. 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)

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

3.2. 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

### 3.3. 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
```

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
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 4. USING FIBRE CHANNEL DEVICES

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

- lpfc
- qla2xxx
- zfcp

4.1. 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:
   
   ```bash
   multipath -ll
   ```

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

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

**Additional resources**

- multipath(8) man page

4.2. 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:

  ```bash
  $ 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

4.3. 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/target/H: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.

### 4.4. 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 5. 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 configfs
- Setting up the NVMe/RDMA target using nvmetcli
- Configuring an NVMe/RDMA client

5.1. 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.

5.2. SETTING UP AN NVME/RDMA TARGET USING CONFIGFS

Use this procedure to configure an NVMe/RDMA target using `configfs`.

**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/
# cd namespaces/

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

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:ff:ff:ff:ff:ff:ff:ff:ff:ff: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

5.3. 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)
  ```
5.4. 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_cmos: 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 6. 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

6.1. 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.

6.2. 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 DISCSRVC ONLINE

NVME Statistics
LS: Xmt 000000000e Compl 000000000e Abort 00000000
LS XMIT: Err 00000000 CMPL: xb 00000000 Err 00000000
Total FCP Compl 000000000000000ea Issue 0000000000000008ec OutO 0000000000000000000002
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=====
ttype: 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   -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:

```bash
# nvme list
```

<table>
<thead>
<tr>
<th>Node Format</th>
<th>SN</th>
<th>Model</th>
<th>Namespace Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

36
6.3. 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

type: 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:

```sh
# 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:

```sh
# nvme list
Node | SN       | Model                                    | Namespace Usage |
----- | -------- | ---------------------------------------- |---------------- |
| Format | FW Rev |
----------------- | --------------- |------------------------------------------ |----------------- |
- | | /dev/nvme0n1 | 80BglFM7xMjbaAAAAAAAC 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 7. 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.

7.1. 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.

7.2. 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:

   ```bash
   # 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:

     ```bash
     # 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
```

<table>
<thead>
<tr>
<th>Node</th>
<th>SN</th>
<th>Model</th>
<th>Namespace</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Format</td>
<td>FW Rev</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>---------------------</td>
<td>----------------</td>
<td>-----------</td>
<td>---------------</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td></td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>/dev/nvme0n1</td>
<td>a34c4f3a0d6f5cec</td>
<td>Linux</td>
<td>1</td>
<td>250.06 GB /</td>
</tr>
<tr>
<td>250.06 GB</td>
<td>512  B + 0 B 4.18.0-2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>/dev/nvme0n2</td>
<td>a34c4f3a0d6f5cec</td>
<td>Linux</td>
<td>2</td>
<td>250.06 GB /</td>
</tr>
<tr>
<td>250.06 GB</td>
<td>512  B + 0 B 4.18.0-2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

### 7.3. 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
   ```
6. Back up the initramfs file system:

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

7. Rebuild the initramfs file system:

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

**Verification**

1. Check that your system recognizes the NVMe devices:

```bash
# nvme list
```

<table>
<thead>
<tr>
<th>Node</th>
<th>SN</th>
<th>Model</th>
<th>Namespace Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Format</td>
<td>FW Rev</td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>---------------------</td>
<td>-------</td>
<td>----------------</td>
</tr>
<tr>
<td>/dev/nvme0n1</td>
<td>a34c4f3a0d6f5cec</td>
<td>Linux</td>
<td>1 250.06 GB /</td>
</tr>
<tr>
<td>250.06 GB</td>
<td>512 B + 0 B 4.18.0-2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>/dev/nvme0n2</td>
<td>a34c4f3a0d6f5cec</td>
<td>Linux</td>
<td>2 250.06 GB /</td>
</tr>
<tr>
<td>250.06 GB</td>
<td>512 B + 0 B 4.18.0-2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>/dev/nvme1n1</td>
<td>a34c4f3a0d6f5cec</td>
<td>Linux</td>
<td>1 250.06 GB /</td>
</tr>
<tr>
<td>250.06 GB</td>
<td>512 B + 0 B 4.18.0-2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>/dev/nvme1n2</td>
<td>a34c4f3a0d6f5cec</td>
<td>Linux</td>
<td>2 250.06 GB /</td>
</tr>
<tr>
<td>250.06 GB</td>
<td>512 B + 0 B 4.18.0-2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>/dev/nvme2n1</td>
<td>a34c4f3a0d6f5cec</td>
<td>Linux</td>
<td>1 250.06 GB /</td>
</tr>
<tr>
<td>250.06 GB</td>
<td>512 B + 0 B 4.18.0-2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>/dev/nvme2n2</td>
<td>a34c4f3a0d6f5cec</td>
<td>Linux</td>
<td>2 250.06 GB /</td>
</tr>
<tr>
<td>250.06 GB</td>
<td>512 B + 0 B 4.18.0-2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>/dev/nvme3n1</td>
<td>a34c4f3a0d6f5cec</td>
<td>Linux</td>
<td>1 250.06 GB /</td>
</tr>
<tr>
<td>250.06 GB</td>
<td>512 B + 0 B 4.18.0-2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>/dev/nvme3n2</td>
<td>a34c4f3a0d6f5cec</td>
<td>Linux</td>
<td>2 250.06 GB /</td>
</tr>
<tr>
<td>250.06 GB</td>
<td>512 B + 0 B 4.18.0-2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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`:

```bash
# nvme list-subsy
```

```
nvme-subsy0 - NQN=testnqn
\+- nvme0 fc traddr=nn-0x20000090fadd5979:pn-0x10000090fadd5979 host_traddr=nn-0x20000090fac7e1de:pn-0x10000090fac7e1dd live
   +- nvme1 fc traddr=nn-0x20000090fadd5979a:pn-0x10000090fadd5979a host_traddr=nn-0x20000090fac7e1de:pn-0x10000090fac7e1dd live
   +- nvme2 fc traddr=nn-0x20000090fadd5979b:pn-0x10000090fadd5979b host_traddr=nn-0x20000090fac7e1de:pn-0x10000090fac7e1dd live
   +- nvme3 fc traddr=nn-0x20000090fadd5979c:pn-0x10000090fadd5979c host_traddr=nn-0x20000090fac7e1de:pn-0x10000090fac7e1dd live
```

```bash
# 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 8. GETTING STARTED WITH SWAP

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

8.1. 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-intense 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.

8.2. 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 8.1. Recommended swap space

<table>
<thead>
<tr>
<th>Amount of RAM in the system</th>
<th>Recommended swap space</th>
<th>Recommended swap space if allowing for hibernation</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;= 2 GB</td>
<td>2 times the amount of RAM</td>
<td>3 times the amount of RAM</td>
</tr>
<tr>
<td>&gt; 2 GB – 8 GB</td>
<td>Equal to the amount of RAM</td>
<td>2 times the amount of RAM</td>
</tr>
<tr>
<td>&gt; 8 GB – 64 GB</td>
<td>At least 4 GB</td>
<td>1.5 times the amount of RAM</td>
</tr>
<tr>
<td>&gt; 64 GB</td>
<td>At least 4 GB</td>
<td>Hibernation not recommended</td>
</tr>
</tbody>
</table>

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.

### 8.3. 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:

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

2. Resize the LVM2 logical volume by 2 GB:

   ```bash
   # 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

8.4. 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:
8.5. 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
```
8.6. 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:
   
   ```bash
   # swapoff -v /dev/VolGroup00/LogVol01
   ```

2. Reduce the LVM2 logical volume by 512 MB:
   
   ```bash
   # lvreduce /dev/VolGroup00/LogVol01 -L -512M
   ```

3. Format the new swap space:
   
   ```bash
   # mkswap /dev/VolGroup00/LogVol01
   ```

4. Activate swap on the logical volume:
   
   ```bash
   # 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:
  
  ```bash
  $ cat /proc/swaps
  $ free -h
  ```

8.7. 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:
   
   ```bash
   # swapoff -v /dev/VolGroup00/LogVol02
   ```

2. Remove the LVM2 logical volume:
   
   ```bash
   # lvremove /dev/VolGroup00/LogVol02
   ```

3. Remove the following associated entry from the /etc/fstab file:
   
   ```bash
   /dev/VolGroup00/LogVol02 swap swap defaults 0 0
   ```

4. Regenerate mount units so that your system registers the new configuration:
8.8. 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 9. 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.

9.1. 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.

9.2. 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:

   ```shell
   # 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:

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

3. Enable and start the `fcoe` service:

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

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

   ```shell
   # 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:

   ```shell
   # 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:

   ```shell
   # 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 10. 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.

10.1. 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.

10.2. 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:
  ```bash
  # dnf install mt-st
  ```

Additional resources

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

10.3. 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:

   ```bash
   # 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
```
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

10.4. 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:

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

Procedure

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

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

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

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

3. Append the data on the tape device:

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

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

   ```bash
   # 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:

```bash
# 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

## 10.5. 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:

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

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

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

- To go to the previous record:

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

- To go to the forward record:

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

Additional resources
10.6. 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`:
    ```bash
    # tar -xzf /dev/st0 /source/directory/
    ```

- For non-rewinding tape devices `/dev/nst0`:
  - Rewind the tape device:
    ```bash
    # mt -f /dev/nst0 rewind
    ```
  - Restore the `etc` directory:
    ```bash
    # tar -xz /dev/nst0 /source/directory/
    ```

**Additional resources**

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

10.7. 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:
   ```bash
   # mt -f /dev/st0 erase
   ```
2. Unload the tape device:

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

Additional resources

- `mt(1)` man page

### 10.8. TAPE COMMANDS

The following are the common `mt` commands:

**Table 10.1. mt commands**

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