Red Hat Enterprise Linux 9.0 Beta

Managing storage devices

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

Last Updated: 2021-11-15

Red Hat Enterprise Linux 9.0 Beta Managing storage devices

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

Legal Notice

Abstract

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

Table of Contents

RHEL BETA RELEASE

Red Hat provides Red Hat Enterprise Linux Beta access to all subscribed Red Hat accounts. The purpose of Beta access is to:

- Provide an opportunity to customers to test major features and capabilities prior to the general availability release and provide feedback or report issues.
- Provide Beta product documentation as a preview. Beta product documentation is under development and is subject to substantial change.

Note that Red Hat does not support the usage of RHEL Beta releases in production use cases. For more information, see What does Beta mean in Red Hat Enterprise Linux and can I upgrade a RHEL Beta installation to a General Availability (GA) release?.

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 input on our documentation. Please let us know how we could make it better. To do so:

- 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 more complex feedback, create a Bugzilla 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. 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.

1.1. INSTALLING TARGETCLI

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

Procedure

1. Install the **targetcli** tool:

```
# yum 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

1.2. 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

1.3. 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

1.4. 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 backstores directory:

```
/> backstores/
```

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> ls
```

Additional resources

- **targetcli(8)** man page

1.5. 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 backstores directory:

```
/> backstores/
```

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> ls
```

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

Additional resources

- **targetcli(8)** man page

1.6. 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 backstores directory:

```
/> backstores/
```

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> ls
```

Additional resources

- **targetcli(8)** man page

1.7. 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 backstores directory:

```
/> backstores/
```

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> ls
```

Additional resources

- **targetcli(8)** man page

1.8. 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

1.9. 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 link:Creating a read-only iSCSI LUN .

3. Configure ACLs. For more information, see Creating an iSCSI ACL .

Additional resources

- **targetcli(8)** man page

1.10. 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. 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/iqn.2006-04.example:444/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.
```

3. Verify the created LUN:

```
/> ls

o- / ...................................................... [...]
  o- backstores ......................................... [...]
  <snip>
  o- iscsi ....................................... [Targets: 1]
  | o- iqn.2006-04.example:444 ................. [TPGs: 1]
  |   o- tpg1 ........................... [no-gen-acls, no-auth]
  |     o- acls ..................................... [ACLs: 2]
  |     | o- 2006-04.com.example.foo:888 .. [Mapped LUNs: 2]
  |     | | o- mapped_lun0 ............. [lun0 block/disk1 (rw)]
  |     | | o- mapped_lun1 ............. [lun1 block/disk2 (ro)]
  |     o- luns ..................................... [LUNs: 2]
  |     | o- lun0 ..................... [block/disk1 (/dev/vdb)]
  |     | o- lun1 ..................... [block/disk2 (/dev/vdc)]
  <snip>
```

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

4. Configure ACLs. For more information, see Creating an iSCSI ACL .

Additional resources

- **targetcli(8)** man page

1.11. 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

1.12. 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

1.13. 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 2. 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.

2.1. 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:

   ```
   # yum 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

2.2. 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

2.3. 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:

```
yum 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

2.4. 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 **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 3. USING FIBRE CHANNEL DEVICES

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

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

3.1. 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

3.2. 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

3.3. 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**/**host*H***/

- **port_id**
- **node_name**
- **port_name**
- **issue_lip**
  Instructs the driver to rediscover remote ports.

3.4. 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 **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. 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

4.1. 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.

4.2. 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/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:ff:12:40:1b:ff:ff:00:00:00:00:00:00: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

4.3. 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:

```
# yum install nvmetcli
```

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

```
# wget
http://git.infradead.org/users/hch/nvmetcli.git/blob_plain/0a6b088db2dc2e5de11e6f23f1e890e4
b54fee64:/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

4.4. 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:

```
# yum 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 5. 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

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. 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:

```
# yum 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 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

5.3. 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:

```
# yum 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_su
bsystem_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_su
bsystem_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_subsysteı* 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 6. GETTING STARTED WITH SWAP

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

6.1. 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.

6.2. 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 6.1. 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 installation*. When prompted to mount the file system, select **Skip**.

6.3. 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
```

6.4. 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
```

6.5. 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
  ```

6.6. 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
```

6.7. 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
```

6.8. 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
```