Oracle® Linux

Gluster Storage for Oracle Linux User's Guide

F18418-03
September 2019

Oracle Legal Notices

Table of Contents

Preface

This document contains information about Gluster Storage for Oracle Linux Release 5. It describes the differences from the upstream version, includes notes on installing and configuring Gluster Storage for Oracle Linux, and provides a statement of what is supported.

Document generated on: 2019-09-26 (revision: 8432)

Audience

This document is written for system administrators and developers who want to use Gluster Storage for Oracle Linux. It is assumed that readers have a general understanding of the Oracle Linux operating system and Gluster storage concepts.

Related Documents

The latest version of this document and other documentation for this product are available at:

Oracle® Linux Documentation

Conventions

The following text conventions are used in this document:

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

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Chapter 1 Introduction to Gluster Storage for Oracle Linux

Gluster is a scalable, distributed file system that aggregates disk storage resources from multiple servers into a single global namespace. This chapter provides introductory information about Gluster Storage for Oracle Linux Release 5.

1.1 About Gluster Storage for Oracle Linux

Gluster Storage for Oracle Linux Release 5 is based on the stable release of the upstream Gluster 5.

Differences between Oracle's version of the software and upstream releases are limited to Oracle specific fixes and patches for specific bugs.

Note

The source RPMs for Gluster are available from Oracle Linux yum server at https://yum.oracle.com.

For comprehensive Gluster documentation, see https://docs.gluster.org/en/latest/.

For more information about Gluster, go to https://www.gluster.org/.

1.2 Notable Updates and New Features

This section contains information on the notable updates and new features in the major releases of Gluster Storage for Oracle Linux.

1.2.1 Notable Updates and New Features in Release 5

New features and bug fixes in the upstream release of Gluster between Release 4.1 and 5 are available in the upstream documentation at:

https://docs.gluster.org/en/latest/release-notes/5.0/

In addition to the upstream changes, the following notable features are included in this release:

- **Gluster block storage.** Gluster volumes can be set up as an iSCSI backstore to provide block storage using the `gluster-block` and `tcmu-runner` packages. Files on volumes are exported as block storage (iSCSI LUNs). For more information, see Section 4.1, “Accessing Volumes using iSCSI”, and the upstream documentation at https://github.com/gluster/gluster-block.
- **Heketi scripted cluster automation.** The `heketi` and `heketi-client` packages automate the management of your Gluster cluster. You can provision trusted storage pools and manage volumes using the `heketi-cli` command, and also write your own custom scripts using the API functions exposed by the Heketi service. It is particularly useful for cloud-based deployments where set up steps can be automated without requiring any manual systems administration. For more information, see Chapter 5, *Automating Volume Lifecycle with Heketi*, and the upstream documentation at https://github.com/heketi/heketi.
- **Upgrade.** Upgrade from Gluster Storage for Oracle Linux Release 4.1 and Release 3.12.

1.2.2 Notable Updates and New Features in Release 4.1

New features and bug fixes in the upstream release of Gluster between Release 3.12 and 4.1 are available in the upstream documentation at:

https://docs.gluster.org/en/latest/release-notes/4.1/

In addition to the upstream changes, the following notable features are included in this release:

- **NFS access with NFS-Ganesha.** You can expose volumes using NFS-Ganesha. NFS-Ganesha is a user space file server for the NFS protocol. It provides a FUSE-compatible File System Abstraction Layer (FSAL) to allow access from any NFS client.
- **Upgrade.** Upgrade from Gluster Storage for Oracle Linux Release 3.12.

1.3 Technical Preview Features

This section contains information on technical preview features available in the major releases of Gluster Storage for Oracle Linux.

1.3.1 Technical Preview Features in Release 5

The following items are highlighted as technical preview features in Gluster Storage for Oracle Linux Release 5:

- **GlusterD-2.0.** GlusterD-2.0 (`glusterd2`) is a re-implementation of `glusterd`. `glusterd2` purports to have better consistency, scalability and performance when compared with the current `glusterd`, while also becoming more modular and easing extensibility. `glusterd2` provides a new management daemon, REST API, and REST client application (`glustercli`). For more information, see the upstream documentation at https://github.com/gluster/glusterd2.

1.3.2 Technical Preview Features in Release 4.1

The following items are highlighted as technical preview features in Gluster Storage for Oracle Linux Release 4.1.

- **Heketi scripted cluster automation.** The `heketi` and `heketi-client` packages automate the management of your cluster. You can provision trusted storage pools and manage volumes using the `heketi-cli` command, and also write your own custom scripts using the API functions exposed by the Heketi service. It is particularly useful for cloud-based deployments where setup steps can be automated without requiring any manual systems administration. For more information, you can read the upstream documentation at https://github.com/heketi/heketi.

Chapter 2 Installing Gluster Storage for Oracle Linux

This chapter discusses how to enable the repositories to install the Gluster Storage for Oracle Linux packages, and how to perform an installation of those packages. This chapter also discusses setting up Gluster trusted storage pools, and Transport Layer security (TLS). This chapter also contains information on upgrading from a previous release of Gluster Storage for Oracle Linux.

2.1 Hardware and Network Requirements

Gluster Storage for Oracle Linux does not require specific hardware; however, certain Gluster operations are CPU and memory intensive. The X6 and X7 line of Oracle x86 Servers are suitable to host Gluster nodes. For more information on Oracle x86 Servers, see:

https://www.oracle.com/servers/x86/index.html

Oracle provides support for Gluster Storage for Oracle Linux on 64-bit x86 (x86_64) and 64-bit Arm (aarch64) hardware.

A minimum node configuration is:

- 2 CPU cores
- 2GB RAM
- 1GB Ethernet NIC
- Dedicated storage sized for your data requirements and formatted as an XFS file system

Although a single 1GB Ethernet NIC is the supported minimum requirement per node, Oracle recommends using 2 x 1GB Ethernet NICs in a bonded (802.3ad/LACP) configuration. Due to the high throughput requirements for distributed and network-based storage 10GB or higher NICs are preferred.

A minimum of three nodes are required in a Gluster trusted storage pool. The examples in this guide use three nodes, named `node1`, `node2`, and `node3`. Node names for each of the nodes in the pool must be resolvable on each host. You can achieve this either by configuring DNS correctly, or you can add host entries to the `/etc/hosts` file on each node.

In the examples in this guide, each host is configured with an additional dedicated block storage device at `/dev/sdb`. The block device is formatted with the XFS file system and then mounted at `/data/glusterfs/myvolume/mybrick`.

Your deployment needs may require nodes with a larger footprint. Additional considerations are detailed in the Gluster upstream documentation.

2.2 Operating System Requirements

Gluster Storage for Oracle Linux Release 5 is available on the platforms and operating systems shown in the following table.

Table 2.1 Operating System Requirements

Platform	Operating System Release	Minimum Operating System Maintenance Release	Kernel
x86_64	Oracle Linux 7	Oracle Linux 7 Update 4	Unbreakable Enterprise Kernel Release 5 (UEK R5)

Platform	Operating System Release	Minimum Operating System Maintenance Release	Kernel
			Unbreakable Enterprise Kernel Release 4 (UEK R4) Red Hat Compatible Kernel (RHCK)
aarch64	Oracle Linux 7	Oracle Linux 7 Update 6	UEK R5

2.3 Enabling Access to the Gluster Storage for Oracle Linux Packages

The Gluster Storage for Oracle Linux packages are available on the Oracle Linux yum server in the `ol7_gluster5` repository, or on the Unbreakable Linux Network (ULN) in the `ol7_x86_64_gluster5` channel, however there are also dependencies across other repositories and channels, and these must also be enabled on each system where Gluster is installed.

Enabling Repositories with ULN

If you are registered to use ULN, use the ULN web interface to subscribe the system to the appropriate channels.

To subscribe to the ULN channels:

1. Log in to https://linux.oracle.com with your ULN user name and password.
2. On the Systems tab, click the link named for the system in the list of registered machines.
3. On the System Details page, click **Manage Subscriptions**.
4. On the System Summary page, select each required channel from the list of available channels and click the right arrow to move the channel to the list of subscribed channels. Subscribe the system to the following channels:
   - `ol7_x86_64_gluster5`
   - `ol7_x86_64_addons`
   - `ol7_x86_64_latest`
   - `ol7_x86_64_optional_latest`
   - `ol7_x86_64_UEKR5` or `ol7_x86_64_UEKR4`
5. Click **Save Subscriptions**.

Enabling Repositories with the Oracle Linux Yum Server

If you are using the Oracle Linux yum server for system updates, enable the Gluster Storage for Oracle Linux yum repository.

To enable the yum repositories:

1. To enable the required repositories on the Oracle Linux yum server, make sure your system is using the modular yum repository configuration. If your system is not using the modular yum

repository configuration, install the `oraclelinux-release-el7` package and run the `/usr/bin/ol_yum_configure.sh` script.

```
# yum install oraclelinux-release-el7
# /usr/bin/ol_yum_configure.sh
```

2. Install the `oracle-gluster-release-el7` release package to install the Gluster Storage for Oracle Linux yum repository configuration.

```
# yum install oracle-gluster-release-el7
```

3. Enable the following yum repositories:

   - `ol7_gluster5`
   - `ol7_addons`
   - `ol7_latest`
   - `ol7_optional_latest`
   - `ol7_UEKR5` or `ol7_UEKR4`

   Use the `yum-config-manager` tool to enable the yum repositories:

```
# yum-config-manager --enable ol7_gluster5 ol7_addons ol7_latest ol7_optional_latest ol7_UEKR5
```

2.4 Installing and Configuring Gluster

A Gluster deployment consists of several systems, known as *nodes*. The nodes form a *trusted storage pool* or *cluster*. Each node in the pool must:

- Have the Oracle Linux operating system installed
- Have the same storage configured
- Have synchronized time
- Be able to resolve the fully qualified domain name of each node in the pool
- Run the Gluster server daemon

The following sections discuss setting up nodes for a Gluster trusted storage pool.

2.4.1 Preparing Oracle Linux Nodes

There are some basic requirements for each Oracle Linux system that you intend to use as a node. These include the following items, for which some preparatory work may be required before you can begin your deployment.

To prepare Oracle Linux nodes:

1. Gluster requires a dedicated file system on each node for the cluster data. The storage must be formatted with an XFS file system. The storage device can be an additional disk, a disk partition, an LVM volume, a loopback device, a multipath device, or a LUN. Do not use the root partition for cluster data.

   The cluster file system used in the examples in this guide is an XFS file system on a disk attached to `/dev/sdb` on each node. This disk is mounted on the directory `/data/glusterfs/myvolume/`

`mybrick`. The inode size is set to 512 bytes to accommodate the extended attributes used by the Gluster file system. To set up this disk, you would use commands similar to:

```
# mkfs.xfs -f -i size=512 -L glusterfs /dev/sdb
# mkdir -p /data/glusterfs/myvolume/mybrick
# echo 'LABEL=glusterfs /data/glusterfs/myvolume/mybrick xfs defaults  0 0' >> /etc/fstab
# mount -a
```

2. Time must be accurate and synchronized across the nodes in the pool. This is achieved by installing and configuring NTP on each node. If the NTP service is not already configured, install and start it. For more information on configuring NTP, see *Oracle® Linux 7: Administrator's Guide*.

3. Pool network communications must be able to take place between nodes within the cluster. If firewall software is running on any of the nodes, it must either be disabled or, preferably, configured to facilitate network traffic on the required ports or between each node on the cluster.

   - If you have a dedicated network for Gluster traffic, you can add the interfaces to a trusted firewall zone and allow all traffic between nodes within the pool. For example, on each node in the pool, run:

     ```
     # firewall-cmd --permanent --change-zone=eno2 --zone=trusted
     # firewall-cmd --reload
     ```

     This command automatically updates the `/etc/sysconfig/network-scripts/ifcfg-eno2` file for the network interface named *eno2*, to add the line `zone=trusted`. You must reload the firewall service for the change to be loaded into the firewall and for it to become active.

     In this configuration, your clients must either be on the same dedicated network and configured for the same firewall zone, or you may need to configure other rules specific to the interface that your clients are connecting on.

   - If your network interfaces are on a shared or untrusted network, you can configure the firewall to allow traffic on the ports specifically used by Gluster:

     ```
     # firewall-cmd --permanent --add-service=glusterfs
     # firewall-cmd --reload
     ```

     Note that adding the `glusterfs` service only exposes the ports required for Gluster. If you intend to add access via Samba, you must add these services as well.

4. All nodes must be able to resolve the fully qualified domain name for each node within the pool. You may either use DNS for this purpose, or provide entries within `/etc/hosts` for each system. If you rely on DNS, it must have sufficient redundancy to ensure that the cluster is able to perform name resolution at any time. If you want to edit the `/etc/hosts` file on each node, add entries for the IP address and host name of all of the nodes in the pool, for example:

   ```
   192.168.1.51    node1.example.com     node1
   192.168.1.52    node2.example.com     node2
   192.168.1.53    node3.example.com     node3
   ```

You can now install and configure the Gluster server.

2.4.2 Installing the Gluster Server

The Gluster server packages should be installed on each node to be included in a trusted storage pool.

To install the Gluster server packages:

1. Install the `glusterfs-server` package.

   ```
   # yum install glusterfs-server
   ```

2. Start and enable the Gluster server service:

```
# systemctl enable --now glusterd
```

2.5 Creating the Trusted Storage Pool

This section shows you how to create a trusted storage pool. In this example, a pool of three servers is created (`node1`, `node2` and `node3`). You should nominate one of the nodes in the pool as the node on which you perform pool operations. In this example, `node1` is the node on which the pool operations are performed.

To create a trusted storage pool:

1. Add the nodes to the trusted server pool. You do not need to add the node on which you are performing the pool operations. For example:

```
# gluster peer probe node2
# gluster peer probe node3
```

2. You can see the status of each node in the pool using:

```
# gluster peer status
```

3. You can see the nodes in the pool using:

```
# gluster pool list
```

If you need to remove a server from a trusted server pool, use:

```
# gluster peer detach hostname
```

2.6 Setting up Transport Layer Security (TLS)

Gluster supports Transport Layer Security (TLS) using the OpenSSL library to authenticate Gluster nodes and clients. TLS encrypts communication between nodes in the trusted storage pool, and between client systems accessing the pool nodes. This is achieved through the use of private keys and public certificates.

Gluster performs mutual authentication in all transactions. This means that if one side of a connection is configured to use TLS then the other side must use it as well. Every node must either have a copy of the public certificate of every other node in the pool, or it must have a copy of the signing CA certificate that it can use to validate the certificates presented by each of the nodes in the pool. Equally, client systems accessing any node in the pool must have a copy of that node's certificate or the signing CA certificate, and the node needs a copy of a certificate for the accessing client.

TLS is enabled as a setting on the volume and can also be enabled for management communication within the pool.

Configuring TLS for your Gluster deployment is optional but recommended for better security.

In production environments, it is recommended you use certificates that are properly signed by a Certificate Authority (CA). This improves validation security and also reduces the complexity of configuration. However, it is not always practical, particularly if you have numerous clients accessing the pool. This section describes configuration for environments where certificates are signed by a CA and for when certificates are self-signed.

To configure TLS on nodes in a Gluster pool:

1. Generate a private key on each node within the pool. You can do this using the `openssl` tool:

```
# openssl genrsa -out /etc/ssl/glusterfs.key 2048
```

2. Create either a self-signed certificate, or a certificate signing request (CSR) using the key that you have created.

   To use self-signed certificates:

   a. To create a self-signed certificate, do:

   ```
   # openssl req -new -x509 -days 365 -key /etc/ssl/glusterfs.key -out /etc/ssl/glusterfs.pem
   ```

   b. When you have generated a self-signed certificate on each node in the storage pool, concatenate the contents of each of these files into a single file. This file should be written to `/etc/ssl/glusterfs.ca` on each node in the pool. Each node uses this file to validate the certificates presented by other nodes or clients that connect to it. If the public certificate for another participatory node or client is not present in this file, the node is unable to verify certificates and the connections fail.

   To use CA-signed certificates:

   a. If you intend to get your certificate signed by a CA, create a CSR by running:

   ```
   # openssl req -new -sha256 -key /etc/ssl/glusterfs.key -out /etc/ssl/glusterfs.csr
   ```

   b. If you generated a CSR and obtained the signed certificate back from your CA, save this file to `/etc/ssl/glusterfs.pem`.

   c. Save the CA certificate for your CA provider to `/etc/ssl/glusterfs.ca` on each node in the pool. Each node uses this file to validate the certificates presented by other nodes or clients that connect to it. If the public certificate for another participatory node or client cannot be verified by the CA signing certificate, attempts to connect by the client or node fail.

3. Configure TLS encryption for management traffic within the storage pool. To do this, create an empty file at `/var/lib/glusterd/secure-access` on each node in the pool. Do the same on any client system where you intend to mount a volume:

   ```
   # touch /var/lib/glusterd/secure-access
   ```

4. Enable TLS on the I/O path for an existing volume by setting the `client.ssl` and `server.ssl` parameters for that volume. For example, to enable TLS on a volume named `myvolume`, do:

   ```
   # gluster volume set myvolume client.ssl on
   # gluster volume set myvolume server.ssl on
   ```

   These parameters enable TLS validation and encryption on client traffic using the Gluster native client and on communications between nodes within the pool. Note that TLS is not automatically enabled on non-native file sharing protocols such as SMB by changing these settings.

5. Restart the `glusterd` service on each of the nodes where you have enabled secure access for management traffic within the pool for these changes to take effect.

   ```
   # systemctl restart glusterd
   ```

2.7 Upgrading Gluster Storage for Oracle Linux to Release 5

This section discusses upgrading to Gluster Storage for Oracle Linux Release 5 from Releases 4.1 or 3.12.

Before you perform an upgrade, configure the Oracle Linux yum server repositories or ULN channels. For information on setting up access to the repositories or channels, see Section 2.3, “Enabling Access to the Gluster Storage for Oracle Linux Packages”.

Make sure you also disable the Gluster Storage for Oracle Linux repositories and channels for the previous releases:

- **Release 4.1.** `ol7_gluster41` repository or `ol7_x86_64_gluster41` ULN channel.
- **Release 3.12.** `ol7_gluster312` repository or `ol7_x86_64_gluster312` ULN channel.

Do not make any configuration changes during the upgrade. You should upgrade the servers before clients are upgraded. After the upgrade, you should run the same Gluster server and client versions.

2.7.1 Performing an Online Upgrade

This procedure performs an online upgrade. An online upgrade does not require any volume down time. During the upgrade, Gluster clients can continue access the volumes.

You can perform an online upgrade with replicated and distributed replicated volumes only. Any other volume types must be upgraded offline. See Section 2.7.2, “Performing an Offline Upgrade” for information on performing an offline upgrade.

This procedure upgrades one server at a time, while keeping the volumes online and client IO ongoing. This procedure assumes that multiple replicas of a replica set are not part of the same server in the trusted storage pool.

The upgrade procedure should be performed on each Gluster node.

Performing an online upgrade:

1. Stop the Gluster service.

```
# systemctl stop glusterd
```

2. Stop all Gluster file system processes:

```
# killall glusterfs glusterfsd
```

   You can make sure no Gluster file system processes are running using:

```
# ps aux |grep gluster
```

3. Stop any Gluster-related services, for example, stop Samba and NFS-Ganesha.

```
# systemctl stop smb
# systemctl stop nfs-ganesha
```

4. Update the Gluster Storage for Oracle Linux packages:

```
# yum update glusterfs-server
```

5. (Optional) If you are using NFS-Ganesha, upgrade the package using:

```
# yum update nfs-ganesha-gluster
```

6. Start the Gluster service:

```
# systemctl daemon-reload
# systemctl start glusterd
```

7. Reload and start any Gluster-related services, for example, Samba and NFS-Ganesha.

```
# systemctl daemon-reload
```

```
# systemctl start smb
# systemctl start nfs-ganesha
```

8. Heal the volumes. You can see the status of the volumes using:

```
# gluster volume status
```

If any bricks in the volume are offline, bring the bricks online using:

```
# gluster volume start volume_name force
```

When all bricks are online, heal the volumes:

```
# for i in `gluster volume list`; do gluster volume heal $i; done
```

You can view healing information for each volume using:

```
# gluster volume heal volume_name info
```

2.7.2 Performing an Offline Upgrade

This procedure performs an offline upgrade. An offline upgrade requires volume down time. During the upgrade, Gluster clients cannot access the volumes. Upgrading the Gluster nodes can be done in parallel to minimise volume down time.

Performing an offline upgrade:

1. Stop any volumes, for example:

```
# gluster volume stop myvolume
```

2. Upgrade all Gluster nodes using the steps provided in Section 2.7.1, “Performing an Online Upgrade”.

Note

You do not need to perform the final step in the online upgrade procedure, which heals the volumes. As the volumes are taken offline during the upgrade, no volume healing is required.

3. Start any volumes, for example:

```
# gluster volume start myvolume
```

2.7.3 Post Upgrade Requirements

This section contains information on steps you should perform after upgrading the nodes in your cluster. You should perform these steps after you have performed either an online or an offline upgrade.

To complete the upgrade:

1. Set the Gluster operating version number for all volumes. You can see the current version setting for all volumes using:

```
# gluster volume get all cluster.op-version
Option                                  Value
------                                  -----
cluster.op-version                      50400
```

If this is not set to `50400`, set it using:

```
# gluster volume set all cluster.op-version 50400
```

2. Upgrade the clients that access the volumes. See Section 2.7.4, “Upgrading Gluster Clients” for information on upgrading Gluster clients.

3. (Optional) For any replicated volumes, you should turn off usage of MD5 checksums during volume healing. This enables you to run Gluster on FIPS-compliant systems.

```
# gluster volume set myvolume fips-mode-rchecksum on
```

2.7.4 Upgrading Gluster Clients

When the Gluster server nodes have been upgraded, you should upgrade any Gluster clients.

To upgrade Gluster clients:

1. Unmount all Gluster mount points on the client.

2. Stop all applications that access the volumes.

3. For Gluster native client (FUSE) clients, update:

```
# yum update glusterfs glusterfs-fuse
```

4. Mount all Gluster shares.

5. Start any applications that were stopped for the upgrade.

Chapter 3 Creating and Managing Volumes

This chapter discusses Gluster volume types and how to create, manage and monitor volumes.

3.1 Creating Volumes

On each node in the trusted storage pool, storage should be allocated for volumes. In the examples in this guide, a file system is mounted on `/data/glusterfs/myvolume/mybrick` on each node. For information on setting up storage on nodes, see Section 2.4.1, “Preparing Oracle Linux Nodes”. Gluster creates a volume on this file system to use as bricks.

There are number of volume types you can use:

- **Distributed:** Distributes files randomly across the bricks in the volume. You can use distributed volumes where the requirement is to scale storage and the redundancy is not required, or is provided by other hardware/software layers. Disk/server failure can result in a serious loss of data as it is spread randomly across the bricks in the volume.
- **Replicated:** Replicates files across bricks in the volume. You can use replicated volumes when high-availability is required.
- **Distributed Replicated:** Distributes files across replicated bricks in the volume. You can use distributed replicated volumes to scale storage and for high-availability and high-reliability. Distributed replicated volumes offer improved read performance.
- **Dispersed:** Provides space efficient protection against disk or server failures (based on erasure codes). This volume type stripes the encoded data of files, with some redundancy added, across multiple bricks in the volume. Dispersed volumes provide a configurable level of reliability with minimum space waste.
- **Distributed Dispersed:** Distributes files across dispersed bricks in the volume. This has the same advantages of distributed replicated volumes, using dispersed instead of replicated to store the data to bricks.

The generally accepted naming convention for creating bricks and volumes is:

```
/data/glusterfs/volume_name/brick_name/brick
```

In this example, *`brick_name`* is the file system that can be mounted from a client. For information on mounting a Gluster file system, see Chapter 4, *Accessing Volumes*.

This section describes the basic steps to set up each of these volume types. When creating volumes, you should include all nodes in the trusted storage pool, including the node on which you are performing the step to create the volume.

The notation used in the examples to create and manage volumes may be provided in the Bash *brace expansion* notation. For example:

```
node{1,2,3}:/data/glusterfs/myvolume/mybrick/brick
```

This is equivalent to providing the node information in the longer form of:

```
node1:/data/glusterfs/myvolume/mybrick/brick node2:/data/glusterfs/myvolume/
mybrick/brick node3:/data/glusterfs/myvolume/mybrick/brick
```

When a volume is configured, you can enable TLS on the volume to authenticate and encrypt connections between nodes that serve data for the volume, and for client systems that connect to the pool to access the volume. See Section 2.6, “Setting up Transport Layer Security (TLS)” for more information.

For more detailed information, see the Gluster upstream documentation.

3.1.1 Creating Distributed Volumes

This section provides an example of creating a pool using a distributed volume.

Example 3.1 Creating a distributed volume

This example creates a distributed volume over three nodes, with one brick on each node.

```
# gluster volume create myvolume node{1,2,3}:/data/glusterfs/myvolume/mybrick/brick
volume create: myvolume: success: please start the volume to access data
# gluster volume start myvolume
volume start: myvolume: success
# gluster volume info

Volume Name: myvolume
Type: Distribute
Volume ID: ...
Status: Started
Snapshot Count: 0
Number of Bricks: 3
Transport-type: tcp
Bricks:
Brick1: node1:/data/glusterfs/myvolume/mybrick/brick
Brick2: node2:/data/glusterfs/myvolume/mybrick/brick
Brick3: node3:/data/glusterfs/myvolume/mybrick/brick
Options Reconfigured:
transport.address-family: inet
nfs.disable: on
```

3.1.2 Creating Replicated Volumes

This section discusses creating a pool using replicated volumes. The `replica` count sets the number of copies of files across bricks in the volume. Generally, two or three copies are used. To protect against server and disk failures, the bricks of the volume should be on different nodes.

Split-brain is a situation where two or more replicated copies of a file become divergent, and there is not enough information to select a copy as being pristine and to self-heal any bad copies. Split-brain situations occur mostly due to network issues with clients connecting to the files in the volume.

If you set `replica` to be an even number (say, 2), you may encounter split-brain as both bricks think they have the latest and correct version. You can use an odd number for the `replica` count (say, 3), to prevent split-brain.

Using an arbiter brick also enables you to avoid split-brain, yet doesn't require the extra storage required of a `replica 3` volume, which needs to store three copies of the files. An arbiter brick contains metadata about the files (but not the files) on other bricks in the volume, so can be much smaller in size. The last brick in each replica subvolume is used as the arbiter brick, for example, if you use `replica 3 arbiter 1`, every third brick is used as an arbiter brick.

Note

Volumes using an arbiter brick can only be created using the `replica 3 arbiter 1` option.

Example 3.2 Creating a replicated volume

This example creates a replicated volume with one brick on three nodes.

```
# gluster volume create myvolume replica 3 \
```

```
  node{1,2,3}:/data/glusterfs/myvolume/mybrick/brick
volume create: myvolume: success: please start the volume to access data
# gluster volume start myvolume
volume start: myvolume: success
# gluster volume info

Volume Name: myvolume
Type: Replicate
Volume ID: ...
Status: Started
Snapshot Count: 0
Number of Bricks: 1 x 3 = 3
Transport-type: tcp
Bricks:
Brick1: node1:/data/glusterfs/myvolume/mybrick/brick
Brick2: node2:/data/glusterfs/myvolume/mybrick/brick
Brick3: node3:/data/glusterfs/myvolume/mybrick/brick
Options Reconfigured:
transport.address-family: inet
nfs.disable: on
performance.client-io-threads: off
```

Example 3.3 Creating a replicated volume with an arbiter

This example creates a replicated volume with one brick on three nodes, and sets one arbiter brick.

```
# gluster volume create myvolume replica 3 arbiter 1 \
  node{1,2,3}:/data/glusterfs/myvolume/mybrick/brick
volume create: myvolume: success: please start the volume to access data
# gluster volume start myvolume
volume start: myvolume: success
# gluster volume info

Volume Name: myvolume
Type: Replicate
Volume ID: ...
Status: Started
Snapshot Count: 0
Number of Bricks: 1 x (2 + 1) = 3
Transport-type: tcp
Bricks:
Brick1: node1:/data/glusterfs/myvolume/mybrick/brick
Brick2: node2:/data/glusterfs/myvolume/mybrick/brick
Brick3: node3:/data/glusterfs/myvolume/mybrick/brick (arbiter)
Options Reconfigured:
transport.address-family: inet
nfs.disable: on
performance.client-io-threads: off
```

3.1.3 Creating Distributed Replicated Volumes

This section discusses creating a pool using distributed replicated volumes. The number of bricks should be a multiple of the `replica` count. For example, six nodes with one brick, or three nodes with two bricks on each node.

The order in which bricks are specified affects data protection. Each `replica` count forms a replica set, with all replica sets combined into a volume-wide distribute set. Make sure that replica sets are not on the same node by listing the first brick on each node, then the second brick on each node, in the same order.

Example 3.4 Creating a distributed replicated volume with one brick on six nodes

This example creates a distributed replicated volume with one brick on six nodes.

```
# gluster volume create myvolume replica 3 \
  node{1..6}:/data/glusterfs/myvolume/mybrick/brick
```

```
volume create: myvolume: success: please start the volume to access data
# gluster volume start myvolume
volume start: myvolume: success
# gluster volume info

Volume Name: myvolume
Type: Distributed-Replicate
Volume ID: ...
Status: Started
Snapshot Count: 0
Number of Bricks: 2 x 3 = 6
Transport-type: tcp
Bricks:
Brick1: node1:/data/glusterfs/myvolume/mybrick/brick
Brick2: node2:/data/glusterfs/myvolume/mybrick/brick
Brick3: node3:/data/glusterfs/myvolume/mybrick/brick
Brick4: node4:/data/glusterfs/myvolume/mybrick/brick
Brick5: node5:/data/glusterfs/myvolume/mybrick/brick
Brick6: node6:/data/glusterfs/myvolume/mybrick/brick
Options Reconfigured:
transport.address-family: inet
nfs.disable: on
performance.client-io-threads: off
```

Example 3.5 Creating a distributed replicated volume with one brick on six nodes with an arbiter

This example creates a distributed replicated volume with one brick on six nodes. Each third brick is used as an arbiter brick.

```
# gluster volume create myvolume replica 3 arbiter 1 \
  node{1..6}:/data/glusterfs/myvolume/mybrick/brick
volume create: myvolume: success: please start the volume to access data
# gluster volume start myvolume
volume start: myvolume: success
# gluster volume info

Volume Name: myvolume
Type: Distributed-Replicate
Volume ID: ...
Status: Created
Snapshot Count: 0
Number of Bricks: 2 x (2 + 1) = 6
Transport-type: tcp
Bricks:
Brick1: node1:/data/glusterfs/myvolume/mybrick/brick
Brick2: node2:/data/glusterfs/myvolume/mybrick/brick
Brick3: node3:/data/glusterfs/myvolume/mybrick/brick (arbiter)
Brick4: node4:/data/glusterfs/myvolume/mybrick/brick
Brick5: node5:/data/glusterfs/myvolume/mybrick/brick
Brick6: node6:/data/glusterfs/myvolume/mybrick/brick (arbiter)
Options Reconfigured:
transport.address-family: inet
nfs.disable: on
performance.client-io-threads: off
```

Example 3.6 Creating a distributed replicated volume with two bricks over three nodes

This example creates a distributed replicated volume with two bricks over three nodes.

```
# gluster volume create myvolume replica 3 \
  node{1,2,3}:/data/glusterfs/myvolume/mybrick/brick1 \
  node{1,2,3}:/data/glusterfs/myvolume/mybrick/brick2
volume create: myvolume: success: please start the volume to access data
# gluster volume start myvolume
volume start: myvolume: success
# gluster volume info
```

```
Volume Name: myvolume
Type: Distributed-Replicate
Volume ID: ...
Status: Started
Snapshot Count: 0
Number of Bricks: 2 x 3 = 6
Transport-type: tcp
Bricks:
Brick1: node1:/data/glusterfs/myvolume/mybrick/brick1
Brick2: node2:/data/glusterfs/myvolume/mybrick/brick1
Brick3: node3:/data/glusterfs/myvolume/mybrick/brick1
Brick4: node1:/data/glusterfs/myvolume/mybrick/brick2
Brick5: node2:/data/glusterfs/myvolume/mybrick/brick2
Brick6: node3:/data/glusterfs/myvolume/mybrick/brick2
Options Reconfigured:
transport.address-family: inet
nfs.disable: on
performance.client-io-threads: off
```

Example 3.7 Creating a distributed replicated volume with two bricks over three nodes with an arbiter

This example creates a distributed replicated volume with two bricks over three nodes. Each third brick is used as an arbiter brick.

```
# gluster volume create myvolume replica 3 arbiter 1 \
  node{1,2,3}:/data/glusterfs/myvolume/mybrick/brick1 \
  node{1,2,3}:/data/glusterfs/myvolume/mybrick/brick2
volume create: myvolume: success: please start the volume to access data
# gluster volume start myvolume
volume start: myvolume: success
# gluster volume info

Volume Name: myvolume
Type: Distributed-Replicate
Volume ID: ...
Status: Started
Snapshot Count: 0
Number of Bricks: 2 x (2 + 1) = 6
Transport-type: tcp
Bricks:
Brick1: node1:/data/glusterfs/myvolume/mybrick/brick1
Brick2: node2:/data/glusterfs/myvolume/mybrick/brick1
Brick3: node3:/data/glusterfs/myvolume/mybrick/brick1 (arbiter)
Brick4: node1:/data/glusterfs/myvolume/mybrick/brick2
Brick5: node2:/data/glusterfs/myvolume/mybrick/brick2
Brick6: node3:/data/glusterfs/myvolume/mybrick/brick2 (arbiter)
Options Reconfigured:
transport.address-family: inet
nfs.disable: on
performance.client-io-threads: off
```

3.1.4 Creating Dispersed Volumes

This section discusses creating a pool using dispersed volumes.

You set the volume redundancy level when you create a dispersed volume. The `redundancy` value sets how many bricks can be lost without interrupting the operation of the volume. The `redundancy` value must be greater than `0`, and the total number of bricks must be greater than `2*redundancy`. A dispersed volume must have a minimum of three bricks.

All bricks of a disperse set should have the same capacity, otherwise, when the smallest brick becomes full, no additional data is allowed in the disperse set.

Example 3.8 Creating a dispersed volume with one brick on three nodes

This example creates a dispersed volume with one brick on three nodes.

```
# gluster volume create myvolume disperse 3 redundancy 1 \
  node{1,2,3}:/data/glusterfs/myvolume/mybrick/brick
volume create: myvolume: success: please start the volume to access data
# gluster volume start myvolume
volume start: myvolume: success
# gluster volume info

Volume Name: myvolume
Type: Disperse
Volume ID: ...
Status: Started
Snapshot Count: 0
Number of Bricks: 1 x (2 + 1) = 3
Transport-type: tcp
Bricks:
Brick1: node1:/data/glusterfs/myvolume/mybrick/brick
Brick2: node2:/data/glusterfs/myvolume/mybrick/brick
Brick3: node3:/data/glusterfs/myvolume/mybrick/brick
Options Reconfigured:
transport.address-family: inet
nfs.disable: on
```

3.1.5 Creating Distributed Dispersed Volumes

This section discusses creating a pool using distributed dispersed volumes. Distributed dispersed volumes consist of two dispersed subvolumes, which are then distributed. The number of bricks should be a multiple of the `disperse` count, and greater than `0`. As a dispersed volume must have a minimum of three bricks, a distributed dispersed volume must have at least six bricks. For example, six nodes with one brick, or three nodes with two bricks on each node are needed for this volume type.

The order in which bricks are specified affects data protection. Each `disperse` count forms a disperse set, with all disperse sets combined into a volume-wide distribute set. Make sure that disperse sets are not on the same node by listing the first brick on each node, then the second brick on each node, in the same order.

The `redundancy` value is used in the same way as for a dispersed volume.

Example 3.9 Creating a distributed dispersed volume with one brick on six nodes

This example creates a distributed dispersed volume with one brick on six nodes.

```
# gluster volume create myvolume disperse 3 redundancy 1 \
  node{1..6}:/data/glusterfs/myvolume/mybrick/brick
volume create: myvolume: success: please start the volume to access data
# gluster volume start myvolume
volume start: myvolume: success
# gluster volume info

Volume Name: myvolume
Type: Distributed-Disperse
Volume ID: ...
Status: Started
Snapshot Count: 0
Number of Bricks: 2 x (2 + 1) = 6
Transport-type: tcp
Bricks:
Brick1: node1:/data/glusterfs/myvolume/mybrick/brick
Brick2: node2:/data/glusterfs/myvolume/mybrick/brick
Brick3: node3:/data/glusterfs/myvolume/mybrick/brick
```

```
Brick4: node4:/data/glusterfs/myvolume/mybrick/brick
Brick5: node5:/data/glusterfs/myvolume/mybrick/brick
Brick6: node6:/data/glusterfs/myvolume/mybrick/brick
Options Reconfigured:
transport.address-family: inet
nfs.disable: on
```

Example 3.10 Creating a distributed dispersed volume with two bricks on three nodes

This example creates a distributed dispersed volume with two bricks on three nodes.

```
# gluster volume create myvolume disperse 3 redundancy 1 \
  node{1,2,3}:/data/glusterfs/myvolume/mybrick/brick1 \
  node{1,2,3}:/data/glusterfs/myvolume/mybrick/brick2
volume create: myvolume: success: please start the volume to access data
# gluster volume start myvolume
volume start: myvolume: success
# gluster volume info

Volume Name: myvolume
Type: Distributed-Disperse
Volume ID: ...
Status: Started
Snapshot Count: 0
Number of Bricks: 2 x (2 + 1) = 6
Transport-type: tcp
Bricks:
Brick1: node1:/data/glusterfs/myvolume/mybrick/brick1
Brick2: node2:/data/glusterfs/myvolume/mybrick/brick1
Brick3: node3:/data/glusterfs/myvolume/mybrick/brick1
Brick4: node1:/data/glusterfs/myvolume/mybrick/brick2
Brick5: node2:/data/glusterfs/myvolume/mybrick/brick2
Brick6: node3:/data/glusterfs/myvolume/mybrick/brick2
Options Reconfigured:
transport.address-family: inet
nfs.disable: on
```

3.2 Managing Volumes

This section provides some basic volume management operations. For more information on volume management, see the upstream documentation.

3.2.1 Setting Volume Options

There are a number of options you can set to configure and tune volumes. These options are set with:

`gluster volume set volume_name option`

For example, to restrict access to mounting the volume to the IP addresses on a network:

```
# gluster volume set myvolume auth.allow 192.168.10.*
```

Instead of, or as well as, making the whole volume available as a mount point, you can set access to volume subdirectories in the same way. For example:

```
# gluster volume set myvolume auth.allow \
  "/(192.168.10.*),/mysubdir1(192.168.1.*),/mysubdir2(192.168.2.*)"
```

3.2.2 Starting a Volume

To start a volume, use the the command:

```
gluster volume start volume_name
```

3.2.3 Stopping a Volume

To stop a volume, use the the command:

```
gluster volume stop volume_name
```

You are requested to confirm the operation. Enter `y` to confirm that you want to stop the volume.

3.2.4 Self Healing a Replicated Volume

The self-heal daemon runs in the background and diagnoses issues with bricks and automatically initiates a self-healing process every 10 minutes on the files that require healing. To see the files that require healing, use:

```
# gluster volume heal myvolume info
```

You can start a self-healing manually using:

```
# gluster volume heal myvolume
```

To list the files in a volume which are in split-brain state, use:

```
# gluster volume heal myvolume info split-brain
```

See the upstream documentation for the methods available to avoid and recover from split-brain issues.

3.2.5 Expanding a Volume

You can increase the number of bricks in a volume to expand available storage. When expanding distributed replicated and distributed dispersed volumes, you need to add a number of bricks that is a multiple of the `replica` or `disperse` count. For example, to expand a distributed replicated volume with a `replica` count of `2`, you need to add bricks in multiples of 2 (such as 4, 6, 8, and so on).

To expand a volume:

1. Prepare the new node with the same configuration and storage as all existing nodes in the trusted storage pool.
2. Add the node to the pool. For example:

   ```
   $ gluster peer probe node4
   ```

3. Add the brick(s), for example:

   ```
   $ gluster volume add-brick myvolume node4:/data/glusterfs/myvolume/mybrick/brick
   ```

4. Rebalance the volume to distribute files to the new brick(s). For example:

   ```
   # gluster volume rebalance myvolume start
   ```

   You can check the status of the volume rebalance using:

   ```
   # gluster volume rebalance myvolume status
   ```

Example 3.11 Creating a distributed replicated volume and adding a node

This example creates a distributed replicated volume with three nodes and two bricks on each node. The volume is then extended to add a new node with an additional two bricks on the node. Note that when you

add a new node to a replicated volume, you need to increase the `replica` count to the new number of nodes in the pool.

```
# gluster volume create myvolume replica 3 \
  node{1,2,3}:/data/glusterfs/myvolume/mybrick/brick1 \
  node{1,2,3}:/data/glusterfs/myvolume/mybrick/brick2
volume create: myvolume: success: please start the volume to access data
# gluster volume start myvolume
volume start: myvolume: success
# gluster volume info

Volume Name: myvolume
Type: Distributed-Replicate
Volume ID: ...
Status: Started
Snapshot Count: 0
Number of Bricks: 2 x 3 = 6
Transport-type: tcp
Bricks:
Brick1: node1:/data/glusterfs/myvolume/mybrick/brick1
Brick2: node2:/data/glusterfs/myvolume/mybrick/brick1
Brick3: node3:/data/glusterfs/myvolume/mybrick/brick1
Brick4: node1:/data/glusterfs/myvolume/mybrick/brick2
Brick5: node2:/data/glusterfs/myvolume/mybrick/brick2
Brick6: node3:/data/glusterfs/myvolume/mybrick/brick2
Options Reconfigured:
transport.address-family: inet
nfs.disable: on
performance.client-io-threads: off

# gluster peer status
Number of Peers: 2

Hostname: node2
Uuid: ...
State: Peer in Cluster (Connected)

Hostname: node3
Uuid: ...
State: Peer in Cluster (Connected)

# gluster peer probe node4
peer probe: success.
# gluster peer status
Number of Peers: 3

Hostname: node2
Uuid: ...
State: Peer in Cluster (Connected)

Hostname: node3
Uuid: ...
State: Peer in Cluster (Connected)

Hostname: node4
Uuid: ...
State: Peer in Cluster (Connected)

# gluster volume add-brick myvolume replica 4 \
>   node4:/data/glusterfs/myvolume/mybrick/brick1 \
>   node4:/data/glusterfs/myvolume/mybrick/brick2
volume add-brick: success
# gluster volume info

Volume Name: myvolume
Type: Distributed-Replicate
Volume ID: ...
```

```
Status: Started
Snapshot Count: 0
Number of Bricks: 2 x 4 = 8
Transport-type: tcp
Bricks:
Brick1: node1:/data/glusterfs/myvolume/mybrick/brick1
Brick2: node2:/data/glusterfs/myvolume/mybrick/brick1
Brick3: node3:/data/glusterfs/myvolume/mybrick/brick1
Brick4: node4:/data/glusterfs/myvolume/mybrick/brick1
Brick5: node1:/data/glusterfs/myvolume/mybrick/brick2
Brick6: node2:/data/glusterfs/myvolume/mybrick/brick2
Brick7: node3:/data/glusterfs/myvolume/mybrick/brick2
Brick8: node4:/data/glusterfs/myvolume/mybrick/brick2
Options Reconfigured:
transport.address-family: inet
nfs.disable: on
performance.client-io-threads: off
# gluster volume rebalance myvolume start
volume rebalance: myvolume: success: Rebalance on myvolume has been started successfully.
Use rebalance status command to check status of the rebalance process.
ID: ...
# gluster volume rebalance myvolume status
...
volume rebalance: myvolume: success
```

Example 3.12 Adding bricks to nodes in a distributed replicated volume

This example adds two bricks to an existing distributed replicated volume. The steps to create this volume are shown in Example 3.11, “Creating a distributed replicated volume and adding a node”.

```
# gluster volume add-brick myvolume \
  node{1,2,3,4}:/data/glusterfs/myvolume/mybrick/brick3 \
  node{1,2,3,4}:/data/glusterfs/myvolume/mybrick/brick4
volume add-brick: success
# gluster volume info

Volume Name: myvolume
Type: Distributed-Replicate
Volume ID: ...
Status: Started
Snapshot Count: 0
Number of Bricks: 4 x 4 = 16
Transport-type: tcp
Bricks:
Brick1: node1:/data/glusterfs/myvolume/mybrick/brick1
Brick2: node2:/data/glusterfs/myvolume/mybrick/brick1
Brick3: node3:/data/glusterfs/myvolume/mybrick/brick1
Brick4: node4:/data/glusterfs/myvolume/mybrick/brick1
Brick5: node1:/data/glusterfs/myvolume/mybrick/brick2
Brick6: node2:/data/glusterfs/myvolume/mybrick/brick2
Brick7: node3:/data/glusterfs/myvolume/mybrick/brick2
Brick8: node4:/data/glusterfs/myvolume/mybrick/brick2
Brick9: node1:/data/glusterfs/myvolume/mybrick/brick3
Brick10: node2:/data/glusterfs/myvolume/mybrick/brick3
Brick11: node3:/data/glusterfs/myvolume/mybrick/brick3
Brick12: node4:/data/glusterfs/myvolume/mybrick/brick3
Brick13: node1:/data/glusterfs/myvolume/mybrick/brick4
Brick14: node2:/data/glusterfs/myvolume/mybrick/brick4
Brick15: node3:/data/glusterfs/myvolume/mybrick/brick4
Brick16: node4:/data/glusterfs/myvolume/mybrick/brick4
Options Reconfigured:
transport.address-family: inet
nfs.disable: on
performance.client-io-threads: off.
# gluster volume rebalance myvolume start
volume rebalance: myvolume: success: Rebalance on myvolume has been started successfully.
Use rebalance status command to check status of the rebalance process.
```

```
ID: ...
# gluster volume rebalance myvolume status
...
volume rebalance: myvolume: success
```

3.2.6 Shrinking a Volume

You can decrease the number of bricks in a volume. This may be useful if a node in the Gluster pool encounters a hardware or network fault.

When shrinking distributed replicated and distributed dispersed volumes, you need to remove a number of bricks that is a multiple of the `replica` or `stripe` count. For example, to shrink a distributed replicate volume with a `replica` count of `2`, you need to remove bricks in multiples of 2 (such as 4, 6, 8, and so on). The bricks you remove must be from the same replica or disperse set.

To shrink a volume:

1. Remove the brick(s), for example:

   ```
   # gluster volume remove-brick myvolume node4:/data/glusterfs/myvolume/mybrick/brick start
   ```

   The `start` option automatically triggers a volume rebalance operation to migrate data from the removed brick(s) to other bricks in the volume.

2. You can check the status of the brick removal using:

   ```
   # gluster volume remove-brick myvolume status
   ```

3. When the `brick-removal` status is `completed`, commit the remove-brick operation. For example:

   ```
   # gluster volume remove-brick myvolume commit
   ```

   You are requested to confirm the operation. Enter `y` to confirm that you want to delete the brick(s).

   The data on the brick is migrated to other bricks in the pool. The data on the removed brick is no longer accessible at the Gluster mount point. Removing the brick removes the configuration information and not the data. You can continue to access the data directly from the brick if required.

Example 3.13 Removing a node from a distributed replicated volume

This example removes a node from a pool with four nodes. The `replica` count for this volume is `4`. As a node is removed, the `replica` count must be reduced to `3`. The `start` option is not needed in replicated volumes, instead, you should use the `force` option. The `force` option means you do not need to check the `remove-brick` process status, or perform the `remove-brick` commit steps.

```
# gluster volume info

Volume Name: myvolume
Type: Distributed-Replicate
Volume ID: ...
Status: Started
Snapshot Count: 0
Number of Bricks: 4 x 4 = 16
Transport-type: tcp
Bricks:
Brick1: node1:/data/glusterfs/myvolume/mybrick/brick1
Brick2: node2:/data/glusterfs/myvolume/mybrick/brick1
Brick3: node3:/data/glusterfs/myvolume/mybrick/brick1
Brick4: node4:/data/glusterfs/myvolume/mybrick/brick1
Brick5: node1:/data/glusterfs/myvolume/mybrick/brick2
Brick6: node2:/data/glusterfs/myvolume/mybrick/brick2
Brick7: node3:/data/glusterfs/myvolume/mybrick/brick2
Brick8: node4:/data/glusterfs/myvolume/mybrick/brick2
```

```
Brick9: node1:/data/glusterfs/myvolume/mybrick/brick3
Brick10: node2:/data/glusterfs/myvolume/mybrick/brick3
Brick11: node3:/data/glusterfs/myvolume/mybrick/brick3
Brick12: node4:/data/glusterfs/myvolume/mybrick/brick3
Brick13: node1:/data/glusterfs/myvolume/mybrick/brick4
Brick14: node2:/data/glusterfs/myvolume/mybrick/brick4
Brick15: node3:/data/glusterfs/myvolume/mybrick/brick4
Brick16: node4:/data/glusterfs/myvolume/mybrick/brick4
Options Reconfigured:
transport.address-family: inet
nfs.disable: on
performance.client-io-threads: off
# gluster volume remove-brick myvolume replica 3 \
  node4:/data/glusterfs/myvolume/mybrick/brick1 \
  node4:/data/glusterfs/myvolume/mybrick/brick2 \
  node4:/data/glusterfs/myvolume/mybrick/brick3 \
  node4:/data/glusterfs/myvolume/mybrick/brick4 \
  force
# gluster volume info

Volume Name: myvolume
Type: Distributed-Replicate
Volume ID: ...
Status: Started
Snapshot Count: 0
Number of Bricks: 4 x 3 = 12
Transport-type: tcp
Bricks:
Brick1: node1:/data/glusterfs/myvolume/mybrick/brick1
Brick2: node2:/data/glusterfs/myvolume/mybrick/brick1
Brick3: node3:/data/glusterfs/myvolume/mybrick/brick1
Brick4: node1:/data/glusterfs/myvolume/mybrick/brick2
Brick5: node2:/data/glusterfs/myvolume/mybrick/brick2
Brick6: node3:/data/glusterfs/myvolume/mybrick/brick2
Brick7: node1:/data/glusterfs/myvolume/mybrick/brick3
Brick8: node2:/data/glusterfs/myvolume/mybrick/brick3
Brick9: node3:/data/glusterfs/myvolume/mybrick/brick3
Brick10: node1:/data/glusterfs/myvolume/mybrick/brick4
Brick11: node2:/data/glusterfs/myvolume/mybrick/brick4
Brick12: node3:/data/glusterfs/myvolume/mybrick/brick4
Options Reconfigured:
transport.address-family: inet
nfs.disable: on
performance.client-io-threads: off
# gluster peer detach node4
peer detach: success
# gluster peer status
Number of Peers: 2

Hostname: node2
Uuid: ...
State: Peer in Cluster (Connected)

Hostname: node3
Uuid: ...
State: Peer in Cluster (Connected)
```

Example 3.14 Removing bricks from a distributed replicated volume

This example removes two bricks from a distributed replicated volume.

```
# gluster volume info

Volume Name: myvolume
Type: Distributed-Replicate
Volume ID: ...
Status: Started
```

```
Snapshot Count: 0
Number of Bricks: 4 x 3 = 12
Transport-type: tcp
Bricks:
Brick1: node1:/data/glusterfs/myvolume/mybrick/brick1
Brick2: node2:/data/glusterfs/myvolume/mybrick/brick1
Brick3: node3:/data/glusterfs/myvolume/mybrick/brick1
Brick4: node1:/data/glusterfs/myvolume/mybrick/brick2
Brick5: node2:/data/glusterfs/myvolume/mybrick/brick2
Brick6: node3:/data/glusterfs/myvolume/mybrick/brick2
Brick7: node1:/data/glusterfs/myvolume/mybrick/brick3
Brick8: node2:/data/glusterfs/myvolume/mybrick/brick3
Brick9: node3:/data/glusterfs/myvolume/mybrick/brick3
Brick10: node1:/data/glusterfs/myvolume/mybrick/brick4
Brick11: node2:/data/glusterfs/myvolume/mybrick/brick4
Brick12: node3:/data/glusterfs/myvolume/mybrick/brick4
Options Reconfigured:
transport.address-family: inet
nfs.disable: on
performance.client-io-threads: off
# gluster volume remove-brick myvolume \
  node{1,2,3}:/data/glusterfs/myvolume/mybrick/brick3 \
  node{1,2,3}:/data/glusterfs/myvolume/mybrick/brick4 \
  start
volume remove-brick start: success
ID: ...
# gluster volume remove-brick myvolume \
  node{1,2,3}:/data/glusterfs/myvolume/mybrick/brick3 \
  node{1,2,3}:/data/glusterfs/myvolume/mybrick/brick4 \
  status
     Node ...        status  run time in h:m:s
--------- ...  ------------     --------------
localhost ...  completed        0:00:00
node2     ...  completed        0:00:00
node3     ...  completed        0:00:01

# gluster volume remove-brick myvolume \
  node{1,2,3}:/data/glusterfs/myvolume/mybrick/brick3 \
  node{1,2,3}:/data/glusterfs/myvolume/mybrick/brick4 \
  commit
Removing brick(s) can result in data loss. Do you want to Continue? (y/n) y
volume remove-brick commit: success
Check the removed bricks to ensure all files are migrated.
If files with data are found on the brick path, copy them via a gluster mount
point before re-purposing the removed brick.
# gluster volume info

Volume Name: myvolume
Type: Distributed-Replicate
Volume ID: ...
Status: Started
Snapshot Count: 0
Number of Bricks: 2 x 3 = 6
Transport-type: tcp
Bricks:
Brick1: node1:/data/glusterfs/myvolume/mybrick/brick1
Brick2: node2:/data/glusterfs/myvolume/mybrick/brick1
Brick3: node3:/data/glusterfs/myvolume/mybrick/brick1
Brick4: node1:/data/glusterfs/myvolume/mybrick/brick2
Brick5: node2:/data/glusterfs/myvolume/mybrick/brick2
Brick6: node3:/data/glusterfs/myvolume/mybrick/brick2
Options Reconfigured:
transport.address-family: inet
nfs.disable: on
performance.client-io-threads: off
```

3.2.7 Deleting a Volume

Deleting a volume erases all data on the volume. To delete a volume, first stop it, then use the command:

`gluster volume delete volume_name`

You are requested to confirm the operation. Enter `y` to confirm that you want to delete the volume and erase all data.

If you want to reuse the storage, you should remove all directories on each node. For example:

```
# rm -rf /data/glusterfs/myvolume/mybrick/*
```

3.3 Monitoring Volumes

You can monitor volumes to help with performance tuning, planning storage capacity, and troubleshooting.

These are the main commands you use for monitoring volumes:

- `gluster volume status`
- `gluster volume profile`
- `gluster volume top`

These commands display information about brick and volume status and performance.

This section contains information on using these monitoring commands.

3.3.1 Using the Volume Status Command

The `gluster volume status` command displays information on the status of bricks and volumes.

To use the command, use the syntax:

`gluster volume status volume_name options`

This section contains some basic examples on how to use the `gluster volume status` command. See the upstream documentation for more information.

Some commands that might be useful are:

`gluster volume status volume_name`	Lists status information for each brick in the volume.
`gluster volume status volume_name detail`	Lists more detailed status information for each brick in the volume.
`gluster volume status volume_name clients`	Lists the clients connected to the volume.
`gluster volume status volume_name mem`	Lists the memory usage and memory pool details for each brick in the volume.
`gluster volume status volume_name inode`	Lists the inode tables of the volume.

`gluster volume status` *`volume_name`* `fd` — Lists the open file descriptor tables of the volume.

`gluster volume status` *`volume_name`* `callpool` — Lists the pending calls for the volume.

Some more detailed examples that include output follow.

Example 3.15 Showing status information about bricks in a volume

This example displays status information about bricks in a volume.

```
# gluster volume status myvolume
Status of volume: myvolume
Gluster process                                 TCP Port  RDMA Port  Online  Pid
------------------------------------------------------------------------------
Brick node1:/data/glusterfs/myvolume/mybric
k/brick                                         49154     0          Y       13553
Brick node2:/data/glusterfs/myvolume/mybric
k/brick                                         49154     0          Y       10212
Brick node3:/data/glusterfs/myvolume/mybric
k/brick                                         49152     0          Y       27358
Brick node4:/data/glusterfs/myvolume/mybric
k/brick                                         49152     0          Y       30502
Brick node5:/data/glusterfs/myvolume/mybric
k/brick                                         49152     0          Y       16282
Brick node6:/data/glusterfs/myvolume/mybric
k/brick                                         49152     0          Y       8913
Self-heal Daemon on localhost                   N/A       N/A        Y       13574
Self-heal Daemon on node3                       N/A       N/A        Y       27379
Self-heal Daemon on node5                       N/A       N/A        Y       16303
Self-heal Daemon on node2                       N/A       N/A        Y       10233
Self-heal Daemon on node6                       N/A       N/A        Y       8934
Self-heal Daemon on node4                       N/A       N/A        Y       30523

Task Status of Volume myvolume
------------------------------------------------------------------------------
There are no active volume tasks
```

Example 3.16 Showing detailed status information about bricks in a volume

This example displays more detailed status information about bricks in a volume.

```
# gluster volume status myvolume detail
Status of volume: myvolume
------------------------------------------------------------------------------
Brick                : Brick node1:/data/glusterfs/myvolume/mybrick/brick
TCP Port             : 49154
RDMA Port            : 0
Online               : Y
Pid                  : 13553
File System          : xfs
Device               : /dev/vdb
Mount Options        : rw,relatime,attr2,inode64,noquota
Inode Size           : N/A
Disk Space Free      : 98.9GB
Total Disk Space     : 100.0GB
Inode Count          : 104857600
Free Inodes          : 104857526
------------------------------------------------------------------------------
...
Brick                : Brick node6:/data/glusterfs/myvolume/mybrick/brick
TCP Port             : 49152
RDMA Port            : 0
Online               : Y
```

```
Pid                  : 8913
File System          : xfs
Device               : /dev/vdb
Mount Options        : rw,relatime,attr2,inode64,noquota
Inode Size           : N/A
Disk Space Free      : 99.9GB
Total Disk Space     : 100.0GB
Inode Count          : 104857600
Free Inodes          : 104857574
```

Example 3.17 Showing information about memory usage for bricks in a volume

This example displays information about memory usage for bricks in a volume.

```
# gluster volume status myvolume mem
Memory status for volume : myvolume
----------------------------------------------
Brick : node1:/data/glusterfs/myvolume/mybrick/brick
Mallinfo
--------
Arena    : 9252864
Ordblks  : 150
Smblks   : 11
Hblks    : 9
Hblkhd   : 16203776
Usmblks  : 0
Fsmblks  : 976
Uordblks : 3563856
Fordblks : 5689008
Keepcost : 30848

----------------------------------------------
...
Brick : node6:/data/glusterfs/myvolume/mybrick/brick
Mallinfo
--------
Arena    : 9232384
Ordblks  : 184
Smblks   : 43
Hblks    : 9
Hblkhd   : 16203776
Usmblks  : 0
Fsmblks  : 4128
Uordblks : 3547696
Fordblks : 5684688
Keepcost : 30848

----------------------------------------------
```

3.3.2 Using the Volume Profile Command

The `gluster volume profile` command displays brick I/O information for each File Operation (FOP) for a volume. The information provided by this command helps you identify where bottlenecks may be in a volume.

Note

Turning on volume profiling may affect system performance, so should be used for troubleshooting and performance monitoring only.

To use the command, use the syntax:

`gluster volume profile` *`volume_name options`*

Use the `gluster volume profile -help` command to show the full syntax.

This section contains some basic examples on how to use the `gluster volume profile` command. See the upstream documentation for more information.

Some commands that might be useful are:

`gluster volume profile` *`volume_name`* `start`	Starts the profiling service for a volume.
`gluster volume profile` *`volume_name`* `info`	Displays the profiling I/O information of each brick in a volume.
`gluster volume profile` *`volume_name`* `stop`	Stops the profiling service for a volume.

A more detailed example of using volume profiling follows.

Example 3.18 Using profiling to monitor a volume

This example turns on profiling for a volume, shows the volume profiling information, then turns profiling off. When profiling is started for a volume, two new diagnostic properties are enabled and displayed when you show the volume information (`diagnostics.count-fop-hits` and `diagnostics.latency-measurement`).

```
# gluster volume profile myvolume start
Starting volume profile on myvolume has been successful
# gluster volume info myvolume

Volume Name: myvolume
Type: Distributed-Replicate
Volume ID: ...
Status: Started
Snapshot Count: 0
Number of Bricks: 2 x 3 = 6
Transport-type: tcp
Bricks:
Brick1: node1:/data/glusterfs/myvolume/mybrick/brick
Brick2: node2:/data/glusterfs/myvolume/mybrick/brick
Brick3: node3:/data/glusterfs/myvolume/mybrick/brick
Brick4: node4:/data/glusterfs/myvolume/mybrick/brick
Brick5: node5:/data/glusterfs/myvolume/mybrick/brick
Brick6: node6:/data/glusterfs/myvolume/mybrick/brick
Options Reconfigured:
diagnostics.count-fop-hits: on
diagnostics.latency-measurement: on
transport.address-family: inet
nfs.disable: on
performance.client-io-threads: off
# gluster volume profile myvolume info
Brick: node1:/data/glusterfs/myvolume/mybrick/brick
--------------------------------------------------
Cumulative Stats:
 %-latency   Avg-latency   Min-Latency   Max-Latency   No. of calls         Fop
 ---------   -----------   -----------   -----------   ------------        ----
      0.00       0.00 us       0.00 us       0.00 us            871  RELEASEDIR
      0.17       2.00 us       2.00 us       2.00 us              3     OPENDIR
      3.07      36.67 us      31.00 us      48.00 us              3      LOOKUP
     10.68      95.75 us      15.00 us     141.00 us              4    GETXATTR
     86.08     514.33 us     246.00 us     908.00 us              6     READDIR

    Duration: 173875 seconds
   Data Read: 0 bytes
```

```
Data Written: 0 bytes

Interval 5 Stats:

    Duration: 45 seconds
   Data Read: 0 bytes
Data Written: 0 bytes
...
# gluster volume profile myvolume stop
Stopping volume profile on myvolume has been successful
```

3.3.3 Using the Volume Top Command

The `gluster volume top` command displays brick performance metrics (read, write, file open calls, file read calls, and so on).

To use the command, use the syntax:

`gluster volume top` *`volume_name options`*

Use the `gluster volume top -help` command to show the full syntax.

This section contains some basic examples on how to use the `gluster volume top` command. See the upstream documentation for more information.

Some commands that might be useful are:

`gluster volume top` *`volume_name`* `read`	Lists the files with the highest open calls on each brick in the volume.
`gluster volume top` *`volume_name`* `write`	Lists the files with the highest write calls on each brick in the volume.
`gluster volume top` *`volume_name`* `open`	Lists the files with the highest open calls on each brick in the volume.
`gluster volume top` *`volume_name`* `opendir`	Lists the files with the highest directory read calls on each brick in the volume.

Some more detailed examples that include output follow.

Example 3.19 Showing performance for all bricks in a volume

This example shows how to display the read and the write performance for all bricks in a volume.

```
# gluster volume top myvolume read-perf  bs 2014 count 1024
Brick: node1:/data/glusterfs/myvolume/mybrick/brick
Throughput 1776.34 MBps time 0.0012 secs
Brick: node2:/data/glusterfs/myvolume/mybrick/brick
Throughput 1694.61 MBps time 0.0012 secs
Brick: node6:/data/glusterfs/myvolume/mybrick/brick
Throughput 1640.68 MBps time 0.0013 secs
Brick: node5:/data/glusterfs/myvolume/mybrick/brick
Throughput 1809.07 MBps time 0.0011 secs
Brick: node4:/data/glusterfs/myvolume/mybrick/brick
Throughput 1438.17 MBps time 0.0014 secs
Brick: node3:/data/glusterfs/myvolume/mybrick/brick
Throughput 1464.73 MBps time 0.0014 secs
# gluster volume top myvolume write-perf bs 2014 count 1024
Brick: node1:/data/glusterfs/myvolume/mybrick/brick
Throughput 779.42 MBps time 0.0026 secs
```

```
Brick: node4:/data/glusterfs/myvolume/mybrick/brick
Throughput 759.61 MBps time 0.0027 secs
Brick: node5:/data/glusterfs/myvolume/mybrick/brick
Throughput 763.26 MBps time 0.0027 secs
Brick: node6:/data/glusterfs/myvolume/mybrick/brick
Throughput 736.02 MBps time 0.0028 secs
Brick: node2:/data/glusterfs/myvolume/mybrick/brick
Throughput 751.85 MBps time 0.0027 secs
Brick: node3:/data/glusterfs/myvolume/mybrick/brick
Throughput 713.61 MBps time 0.0029 secs
```

Example 3.20 Showing performance for a brick

This example shows how to display the read and the write performance for a brick.

```
# gluster volume top myvolume read-perf bs 2014 count 1024 brick \
  node1:/data/glusterfs/myvolume/mybrick/brick
Brick: node1:/data/glusterfs/myvolume/mybrick/brick
Throughput 1844.67 MBps time 0.0011 secs
# gluster volume top myvolume write-perf bs 2014 count 1024 brick \
  node1:/data/glusterfs/myvolume/mybrick/brick
Brick: node1:/data/glusterfs/myvolume/mybrick/brick
Throughput 612.88 MBps time 0.0034 secs
```

Chapter 4 Accessing Volumes

This chapter the discusses the options available to access Gluster volumes from an Oracle Linux or Microsoft Windows client system.

Access to volumes is provided through a number of different network file system technologies including NFS, Samba and a Gluster native client that uses the File System in Userspace (FUSE) software interface to provide access to the volume.

If you need to mount the volume locally on one of the nodes, you should treat this as an additional mount exactly as if you were mounting from a remote host.

Warning

Editing the data within the volume directly on the file system on each node can quickly lead to split-brain scenarios and potential file system corruption.

4.1 Accessing Volumes using iSCSI

This section discusses setting up a volume as an iSCSI backstore to provide block storage using the `gluster-block` and `tcmu-runner` packages. Files on volumes are exported as block storage (iSCSI LUNs). The storage initiator logs into the LUN to access the block device.

The `gluster-block` package includes a CLI to create and manage iSCSI access to volumes. The `tcmu-runner` package handles access to volumes using the iSCSI protocol.

4.1.1 Installing iSCSI Services

This section discusses setting up the trusted storage pool to enable iSCSI access.

To install iSCSI services:

On each node in the trusted storage pool:

1. Install the `tcmu-runner` and `gluster-block` packages.

```
# yum install tcmu-runner gluster-block
```

2. Start and enable the `tcmu-runner` and `gluster-blockd` services:

```
# systemctl enable --now tcmu-runner gluster-blockd
```

4.1.2 Creating a Block Device

This section discusses creating a block device on an existing volume. For more information about creating and managing block devices, see the upstream documentation. To get help on using the `gluster-block` command, enter `gluster-block help`.

To create a block device:

On a node in the trusted storage pool:

1. Create the block device using the `gluster-block create` command. This example creates a block device named `myvolume-block` for the volume named `myvolume`. The three nodes in the trusted storage pool form a high availability connection to the volume.

```
# gluster-block create myvolume/myvolume-block ha 3 prealloc no \
   192.168.1.51,192.168.1.52,192.168.1.53  20GiB
IQN: iqn.2016-12.org.gluster-block:4a015741-f455-4568-a0ee-b333b595ba4f
PORTAL(S):  10.147.25.88:3260 10.147.25.89:3260 10.147.25.90:3260
```

```
RESULT: SUCCESS
```

2. To get a list of block devices for a volume, use the `gluster-block list` command.

```
# gluster-block list myvolume
myvolume-block
```

3. You can get information on the block device using the `gluster-block info` command.

```
# gluster-block info myvolume/myvolume-block
NAME: myvolume-block
VOLUME: myvolume
GBID: 4a015741-f455-4568-a0ee-b333b595ba4f
SIZE: 20.0 GiB
HA: 3
PASSWORD:
EXPORTED ON: 192.168.1.51 192.168.1.52 192.168.1.53
```

4. To get a list of the iSCSI targets, use the `targetcli ls` command.

```
# targetcli  ls
...
  o- iscsi ..................................................................... [Targets: 1]
  | o- iqn.2016-12.org.gluster-block:4a015741-f455-4568-a0ee-b333b595ba4f ........ [TPGs: 3]
  |   o- tpg1 ..................................................... [gen-acls, no-auth]
  |   | o- acls ................................................................... [ACLs: 0]
  |   | o- luns ................................................................... [LUNs: 1]
  |   | | o- lun0 ............................... [user/myvolume-block (glfs_tg_pt_gp_ao)]
  |   | o- portals ................................................... [Portals: 1]
  |   |   o- 192.168.1.51:3260 ..................................................... [OK]
...
```

4.1.3 Accessing an iSCSI Block Device

This section discusses accessing an iSCSI block device.

To access an iSCSI block device:

On a client node:

1. Install the packages required to access the block storage.

```
# yum install iscsi-initiator-utils device-mapper-multipath
```

2. Enable the `iscsid` service:

```
# systemctl enable iscsid
```

3. Discover and log into the iSCSI target on any of the nodes in the trusted storage pool that is set to host block devices. For example:

```
# iscsiadm -m discovery -t st -p 192.168.1.51 -l
```

4. You can see a list of the iSCSI sessions using the `iscsiadm -m session` command:

```
# iscsiadm -m session
tcp: [1] 192.168.1.51:3260,1 iqn.2016-12.org.gluster-block:4a015741... (non-flash)
tcp: [2] 192.168.1.52:3260,2 iqn.2016-12.org.gluster-block:4a015741... (non-flash)
tcp: [3] 192.168.1.53:3260,3 iqn.2016-12.org.gluster-block:4a015741... (non-flash)
```

5. (Optional) Set up multipath.

   To set up multipath:

   a. Load and enable the multipath module.

```
# modprobe dm_multipath
# mpathconf --enable
```

b. Restart and enable the `multipathd` service.

```
# systemctl restart multipathd
# systemctl enable multipathd
```

6. To see the new iSCSI devices added, use the `lsblk` command:

```
# lsblk
NAME                MAJ:MIN RM  SIZE RO TYPE  MOUNTPOINT
sdd                   8:48   0   20G  0 disk
└─mpatha            252:2    0   20G  0 mpath
sdb                   8:16   0   10G  0 disk
sde                   8:64   0   20G  0 disk
└─mpatha            252:2    0   20G  0 mpath
sdc                   8:32   0   20G  0 disk
└─mpatha            252:2    0   20G  0 mpath
sda                   8:0    0 36.5G  0 disk
├─sda2                8:2    0   36G  0 part
│ ├─vg_main-lv_swap 252:1    0    4G  0 lvm   [SWAP]
│ └─vg_main-lv_root 252:0    0   32G  0 lvm   /
└─sda1                8:1    0  500M  0 part  /boot
```

New disks are added for the Gluster block storage. In this case, the disks are `sdd`, `sde`, and `sdc`.

7. Create an XFS file system on the device:

```
# mkfs.xfs /dev/mapper/mpatha
```

8. Mount the block device. In this example the Gluster block storage is mounted to `/mnt`.

```
# mount /dev/mapper/mpatha /mnt/
```

4.2 Accessing Volumes using NFS

You can expose volumes using NFS-Ganesha. NFS-Ganesha is a user space file server for the NFS protocol. It provides a FUSE-compatible File System Abstraction Layer (FSAL) to allow access from any NFS client.

To access a volume using NFS:

On each node in the trusted storage pool on which you want to enable NFS access:

1. Install the Gluster NFS-Ganesha client packages:

```
# yum install nfs-ganesha-gluster
```

2. Create an export configuration file in the `/etc/ganesha/exports` directory. This file contains the NFS export information for NFS Ganesha. In this example we use the file name `export.myvolume.conf` to export a volume named `myvolume` to an NFS share located at `/myvolume` on the node.

```
EXPORT{
    Export_Id = 1 ;   # Export ID unique to each export
    Path = "/myvolume";  # Path of the volume to be exported. Eg: "/test_volume"

    FSAL {
        name = GLUSTER;
        hostname = "localhost";  # IP of one of the nodes in the trusted pool
        volume = "myvolume";  # Volume name. Eg: "test_volume"
```

```
    }

    Access_type = RW;    # Access permissions
    Squash = No_root_squash; # To enable/disable root squashing
    Disable_ACL = TRUE;  # To enable/disable ACL
    Pseudo = "/myvolume";  # NFSv4 pseudo path for this export. Eg: "/test_volume_pseudo"
    Protocols = "3","4" ;    # NFS protocols supported
    Transports = "UDP","TCP" ; # Transport protocols supported
    SecType = "sys";     # Security flavors supported
}
```

Edit the `/etc/ganesha/ganesha.conf` file to include the new export configuration file, for example:

```
...
%include "/etc/ganesha/exports/export.myvolume.conf"
```

3. Enable and start the `nfs-ganesha` service:

```
# systemctl enable --now nfs-ganesha
```

Note

If the volume is created *after* you set up access using NFS, you must reload the `nfs-ganesha` service:

```
# systemctl reload-or-restart nfs-ganesha
```

4. Check the volume is exported:

```
# showmount -e localhost
Export list for localhost:
/myvolume (everyone)
```

5. To connect to the volume from an NFS client, mount the NFS share, for example:

```
# mkdir /gluster-storage
# mount node1:/myvolume /gluster-storage
```

Any files created in this `/gluster-storage` directory on the NFS client are written to the volume.

4.3 Accessing Volumes using the Gluster Native Client (FUSE)

You can use the Gluster native client on an Oracle Linux host to access a volume. The native client takes advantage of the File System in Userspace (FUSE) software interface that allows you to mount a volume without requiring a kernel driver or module.

To access a volume using the Gluster native client (FUSE):

1. On the host where you intend to mount the volume, enable access to the Gluster Storage for Oracle Linux packages. For information on enabling access, see Section 2.3, “Enabling Access to the Gluster Storage for Oracle Linux Packages”.

2. Install the Gluster native client packages:

```
# yum install glusterfs glusterfs-fuse
```

3. Create the directory where you intend to mount the volume. For example:

```
# mkdir /gluster-storage
```

4. If you have configured TLS for a volume, you may need to perform additional steps before a client system is able to mount the volume. See Section 2.6, “Setting up Transport Layer Security (TLS)” for more information. The following steps are required to complete client configuration for TLS:

To set up TLS with the Gluster native client (FUSE):

a. Set up a certificate and private key on the client system. You can either use a CA signed certificate or create a self-signed certificate, as follows:

```
# openssl req -newkey rsa:2048 -nodes -keyout /etc/ssl/glusterfs.key \
   -x509 -days 365 -out /etc/ssl/glusterfs.pem
```

b. Append the client certificate to the `/etc/ssl/glusterfs.ca` file on each node in the trusted server pool. Equally, ensure that the client has a copy of the `/etc/ssl/glusterfs.ca` file that includes either the CA certificate that signed each node's certificate, or that contains all of the self-signed certificates for each node. Since Gluster performs mutual authentication, it is essential that both the client and the server node are able to validate each other's certificates.

c. If you enabled encryption on management traffic, you must enable this facility on the client system to allow it to perform the initial mount. To do this, Gluster looks for a file at `/var/lib/glusterfs/secure-access`. This directory may not exist on a client system, so you might need to create it before touching the file:

```
# mkdir -p /var/lib/glusterfs
# touch /var/lib/glusterfs/secure-access
```

d. If the volume is already set up and running before you added the client certificate to `/etc/ssl/glusterfs.ca`, you must stop the volume, restart the Gluster service and start up the volume again for the new certificate to be registered:

```
# gluster volume stop myvolume
# systemctl restart glusterd
# gluster volume start myvolume
```

5. Mount the volume on the directory using the `glusterfs` mount type and by specifying a node within the pool along with the volume name. For example:

```
# mount -t glusterfs node1:myvolume /gluster-storage
```

If you have set up the volume to enable mounting a subdirectory, you can add the subdirectory name to the path on the Gluster file system:

```
# mount -t glusterfs node1:myvolume/subdirectory /gluster-storage
```

6. Check the permissions on the new mount to make sure the appropriate users can read and write to the storage. For example:

```
# chmod 777 /gluster-storage
```

7. To make the mount permanent, edit your `/etc/fstab` file to include the mount. For example:

```
node1:/myvolume /gluster-storage glusterfs defaults,_netdev 0 0
```

If you are mounting a subdirectory on the volume, add the subdirectory name to the path on the Gluster file system. For example:

```
node1:/myvolume/subdirectory /gluster-storage glusterfs defaults,_netdev 0 0
```

If you have trouble mounting the volume, you can check the logs on the client system at `/var/log/glusterfs/` to try to debug connection issues. For example, if TLS is not properly configured and the server node is unable to validate the client, you may see an error similar to the following in the logs:

```
… error:14094418:SSL routines:ssl3_read_bytes:tlsv1 alert unknown ca
```

4.4 Accessing Volumes using Samba

You can expose volumes using the Common Internet File System (CIFS) or Server Message Block (SMB) by using Samba. This file sharing service is commonly used on Microsoft Windows systems.

Gluster provides hooks to preconfigure and export volumes automatically using a Samba Virtual File System (VFS) module plug-in. This reduces the complexity of configuring Samba to export the shares and also means that you do not have to pre-mount the volumes using the FUSE client, resulting in some performance gains. The hooks are triggered every time a volume is started, so your Samba configuration is updated the moment a volume is started within Gluster.

For more information on Samba, see *Oracle® Linux 7: Administrator's Guide*.

Setting up the Volume for Samba Access

This section discusses setting up the nodes in the trusted storage pool to enable access to a volume by a Samba client. To use this service, you must make sure both the `samba` and `samba-vfs-glusterfs` packages are installed on any of the nodes in the pool where you intend a client to connect to the volume using Samba.

To set up the volume for Samba access:

On each node in the trusted storage pool on which you want to enable Samba access:

1. Install the Samba packages for Gluster:

```
# yum install samba samba-vfs-glusterfs
```

2. If you are running a firewall service, enable access to Samba on the node. For example:

```
# firewall-cmd --permanent --add-service=samba
# firewall-cmd --reload
```

3. Enable and start the Samba service:

```
# systemctl enable --now smb
```

4. (Optional) If you do not have an authentication system configured (for example, an LDAP server), you can create a Samba user to enable access to the Samba share from clients. This user should be created on all nodes set up to export the Samba share. For example:

```
# adduser myuser
# smbpasswd -a myuser
New SMB password:
Retype new SMB password:
Added user myuser.
```

   Restart the Samba service:

```
# systemctl restart smb
```

5. (Optional) If you want to allow guest access to a Samba share (no authentication is required), add a new line containing `map to guest = Bad User` to the `[global]` section of the `/etc/samba/smb.conf` file on each node set up to export the Samba share. For example:

```
[global]
       workgroup = SAMBA
       security = user
```

```
        passdb backend = tdbsam

        printing = cups
        printcap name = cups
        load printers = yes
        cups options = raw
        map to guest = Bad User
```

Allowing guest access also requires that the `[gluster-volume_name]` section contains the `guest ok = yes` option, which is set automatically with the Gluster hook scripts in the next step.

Restart the Samba service:

```
# systemctl restart smb
```

6. If you have a running volume, stop it and start it again. On any node in the trusted storage pool, run:

```
# gluster volume stop myvolume
# gluster volume start myvolume
```

When you start a volume, a Gluster hook is triggered to automatically add a configuration entry for the volume to the `/etc/samba/smb.conf` file on each node, and to reload the Samba service. This script generates a Samba configuration entry similar to the following:

```
[gluster-myvolume]
comment = For samba share of volume myvolume
vfs objects = glusterfs
glusterfs:volume = myvolume
glusterfs:logfile = /var/log/samba/glusterfs-myvolume.%M.log
glusterfs:loglevel = 7
path = /
read only = no
guest ok = yes
kernel share modes = no
```

Note

The value of the `[gluster-myvolume]` entry sets the name you use to connect to the Samba share in the connection string.

7. (Optional) If you do not want Gluster to automatically configure Samba to export shares for volumes, you can remove or rename the hook scripts that control this behavior. On each node on which you want to disable the Samba shares, rename the hook scripts, for example:

```
# rename S30 disabled-S30 $(find /var/lib/glusterd -type f -name S30samba*)
```

To re-enable the hooks, you can run:

```
# rename disabled-S30 S30 $(find /var/lib/glusterd -type f -name *S30samba*)
```

Testing SMB Access to a Volume

This section discusses testing SMB access to a volume that has been set up to export a Samba share. You can test SMB access to the volume from an Oracle Linux host. This host does not need to be part of the Gluster pool.

To test access to the volume using SMB:

1. On an Oracle Linux host, install the Samba client package:

```
# yum install samba-client
```

2. Use the `smbclient` command to list Samba shares on a node in the trusted storage pool where you set up Samba. For example:

```
# smbclient -N -U% -L node1
```

   To look directly at the contents of the volume, you can do:

```
# smbclient -N -U% //node1/gluster-myvolume -c ls
```

   In this command, you specify the Samba share name for the volume. This name can be found on a host where you set up the Samba share, in the `/etc/samba/smb.conf` file. Usually the Samba share name is `gluster-volume_name`.

Testing CIFS Access to a Volume

This section discusses testing CIFS access to a volume that has been set up to export a Samba share. You can test CIFS access to the volume from an Oracle Linux host. This host does not need to be part of the Gluster pool.

To test access to the volume using CIFS:

1. On an Oracle Linux host, install the `cifs-utils` package:

```
# yum install cifs-utils
```

2. Create a mount directory where you intend to mount the volume. For example:

```
# mkdir /gluster-storage
```

3. Mount the volume on the directory using the `cifs` mount type and by specifying a node within the pool, along with the Samba share name for the volume. This name can be found on a host where you set up the Samba share, in the `/etc/samba/smb.conf` file. Usually the Samba share name is `gluster-volume_name`. For example:

```
# mount -t cifs -o guest //node1/gluster-myvolume /gluster-storage
```

   If you have set up the volume to enable mounting a subdirectory, you can add the subdirectory name to the path on the Gluster file system:

```
# mount -t cifs -o guest //node1/gluster-myvolume/subdirectory /gluster-storage
```

   If you want to pass authentication credentials to the Samba share, first add them to a local file. In this example, the credentials are saved to the file `/credfile`.

```
username=value
password=value
```

   Set the permissions on the credentials file so other users cannot access it.

```
# chmod 600 /credfile
```

   You can then use the credentials file to connect to the Samba share, for example:

```
# mount -t cifs -o credentials=/credfile //node1/gluster-myvolume /gluster-storage
```

4. Check the permissions on the new mount to make sure the appropriate users can read and write to the storage. For example:

```
# chmod 777 /gluster-storage
```

Accessing the Volume from Microsoft Windows

On a Microsoft Windows host, you can mount the volume using the Samba share. By default, the Samba share is available in the Workgroup named `SAMBA` (as defined in the `/etc/samba/smb.conf` file on Samba share nodes).

You can map the volume by mapping a network drive using Windows Explorer using the format: `\\node\volume`, for example:

```
\\node1\gluster-myvolume
```

Alternatively, you can map a new drive using the Windows command line. Start the **Command Prompt**. Enter a command similar to the following:

```
net use z: \\node1\gluster-myvolume
```

Chapter 5 Automating Volume Lifecycle with Heketi

Heketi is a service that provides a RESTful interface (the Heketi API) for managing the full lifecycle of Gluster Storage for Oracle Linux trusted storage pools and volumes. For example, Heketi can fully automate the steps defined in Section 2.5, “Creating the Trusted Storage Pool” and Section 3.1, “Creating Volumes”. You can write scripts to dynamically create, alter and destroy any clusters and volumes that Heketi manages.

Heketi uses the term *cluster* for Gluster trusted storage pools. This chapter uses the term cluster, which can be interchanged with the term trusted storage pool.

Heketi is especially helpful in managed cloud-based deployments where you can create volumes in a fast, stable and fully-automated way using the Heketi API, without any manual systems administration.

The Heketi client includes a CLI (`heketi-cli`) for creating and managing clusters, nodes, devices, and volumes. Although the Heketi CLI is available, you should use the Heketi API for automated management of clusters and volumes.

The latest Heketi documentation is available upstream at https://github.com/heketi/heketi/tree/master/docs.

5.1 Installing the Heketi API

To set up the Heketi API, install the Heketi service on a node in the proposed cluster (trusted storage pool), or on a separate server that is not part of the cluster.

To install and set up the Heketi API:

1. Prepare the hosts and make sure the `glusterd` service is running on each node to be used in the Heketi cluster.

   Do not create Gluster trusted storage pools or volumes using the `gluster` command.

   Do not format the disks to use for volumes. The disks must be in RAW format to use them with Heketi.

   For information on preparing nodes and installing the `glusterd` service, see Section 2.4, “Installing and Configuring Gluster”.

2. Install the Heketi server on a node in the Heketi cluster, or on a separate server:

   ```
   # yum install heketi
   ```

3. The Heketi server node must have password-less SSH key access to all nodes in the Heketi cluster.

   You can either use the root user on each node in the Heketi cluster to set up SSH access, or you can use a non-root user. If you use a non-root user, set the user up on each node in the cluster, and make sure the user has `sudo` permissions. The user is not required on the Heketi server node unless the server node is also part of the Heketi cluster.

   On the Heketi server node, generate a password-less SSH key. For example:

   ```
   # ssh-keygen -f /mypath/heketi_key -t rsa -N ''
   ```

   Copy the public key to each node in the Heketi cluster. For example:

   ```
   # ssh-copy-id -i /mypath/heketi_key root@node1.example.com
   # ssh-copy-id -i /mypath/heketi_key root@node2.example.com
   # ssh-copy-id -i /mypath/heketi_key root@node3.example.com
   ```

You can test the key has been set up correctly by using it to log into a Heketi cluster node from the Heketi server node. For example:

```
# ssh -i /mypath/heketi_key root@node1.example.com
```

On the Heketi server node, add this user to the `sshexec` section in the Heketi service configuration file, `/etc/heketi/heketi.json`. For example, for the root user with the SSH private key located at `/mypath/heketi_key`:

```
    "_sshexec_comment": "SSH username and private key file information",
    "sshexec": {
      "keyfile": "/mypath/heketi_key",
      "user": "root"
    },
```

4. (Optional) Configure other Heketi service options in the Heketi service configuration file, `/etc/heketi/heketi.json`. For example, set the API service port number (the default is `8080`), or set up user access credentials.

5. Start and enable the `heketi` service:

```
# systemctl enable --now heketi
```

6. You can verify the `heketi` service is running by sending a GET request to the Heketi API:

```
# curl http://localhost:8080/hello
Hello from Heketi
```

5.2 Installing the Heketi Client

The Heketi client package includes a CLI to manage volumes using the Heketi API. The Heketi client should be installed on a client node, not on a host that is part of the Heketi cluster.

```
# yum install heketi-client
```

5.3 Using the Heketi CLI

This section shows you how to create a cluster, and create and manage volumes using the Heketi CLI. The steps in this section should be performed on the node on which you installed the Heketi client.

Heketi cannot retrieve information about an existing cluster. New clusters must be created for them to be managed by Heketi. You can create multiple clusters with various disk types (SSD, SAS, or SATA) to suit your needs.

After Heketi is set up to manage a cluster, only use `heketi-cli` commands or the Heketi API to manage the cluster and volumes. You should not use `gluster` commands to manage clusters or volumes managed by Heketi, as it may cause inconsistencies in the Heketi database.

The RAW devices used by Heketi to create volumes must not be formatted.

The hostnames and IP addresses uses in the examples in this section are:

- node1.example.com (192.168.1.51)
- node2.example.com (192.168.1.52)
- node3.example.com (192.168.1.53)

When you run `heketi-cli` commands, you need to specify the Heketi API server location, and if authentication has been set up, the authentication information. You can do this either by passing those options when running `heketi-cli` commands, or set environment variables. The `heketi-cli` syntax to use is:

`heketi-cli --server=URL --user=username --secret=key command`

If you would prefer to use environment variables, the environment variable names are as shown in this example.

```
# export HEKETI_CLI_SERVER=http://node1.example.com:8080
# export HEKETI_CLI_USER=admin
# export HEKETI_CLI_KEY=key
```

The examples in this section use environment variables to make the commands easier to read.

5.3.1 Creating a Cluster

This section discusses creating a cluster with the Heketi CLI.

To create a cluster:

1. Create Heketi topology configuration file to set up the cluster. Copy the `/usr/share/heketi/topology-sample.json` to a new file, for example:

   ```
   # cp /usr/share/heketi/topology-sample.json /usr/share/heketi/topology-mycluster.json
   ```

2. The topology file is in JSON format, and can contain an array of clusters. Each cluster contains an array of nodes. Each node contains the node hostname and IP address, the devices available on the node, and the failure domain (zone) on which the node should be included.

   Edit the `/usr/share/heketi/topology-mycluster.json` file to to suit your environment. For example:

   ```
   {
     "clusters": [
       {
         "nodes": [
           {
             "node": {
               "hostnames": {
                 "manage": [
                   "node1.example.com"
                 ],
                 "storage": [
                   "192.168.1.51"
                 ]
               },
               "zone": 1
             },
             "devices": [
               {
                 "name": "/dev/sdb",
                 "destroydata": false
               }
             ]
           },
           {
             "node": {
               "hostnames": {
                 "manage": [
                   "node2.example.com"
   ```

```
            ],
            "storage": [
              "192.168.1.52"
            ]
          },
          "zone": 1
        },
        "devices": [
          {
            "name": "/dev/sdb",
            "destroydata": false
          }
        ]
      },
      {
        "node": {
          "hostnames": {
            "manage": [
              "node3.example.com"
            ],
            "storage": [
              "192.168.1.53"
            ]
          },
          "zone": 1
        },
        "devices": [
          {
            "name": "/dev/sdb",
            "destroydata": false
          }
        ]
      }
    ]
  }
 ]
}
```

3. Load the Heketi topology file into Heketi using the `heketi-cli topology load` command to create a cluster. The clusters, nodes and disks specified in the JSON file are created.

```
# heketi-cli topology load --json=heketi-mycluster.json
Creating cluster ... ID: 7c1cf54ff4b5ab41f823ac592ba68ca5
 Allowing file volumes on cluster.
 Allowing block volumes on cluster.
 Creating node node1.example.com ... ID: c35ba48b042555633b511f459f5aa157
  Adding device /dev/sdb ... OK
 Creating node node2.example.com ... ID: 7c899bc9f50e46efc993dc22263549e4
  Adding device /dev/sdb ... OK
 Creating node node3.example.com ... ID: 32755ad123c325f75c91aa963c4312f3
  Adding device /dev/sdb ... OK
```

4. You can get a list of clusters managed by Heketi using the `heketi-cli cluster list` command:

```
# heketi-cli cluster list
Clusters:
Id:7c1cf54ff4b5ab41f823ac592ba68ca5 [file][block]
```

5. You can get more information about each cluster managed by Heketi using the `heketi-cli topology info` command:

```
# heketi-cli topology info

Cluster Id: 7c1cf54ff4b5ab41f823ac592ba68ca5

    File:  true
```

```
   Block: true

   Volumes:

   Nodes:

Node Id: 32755ad123c325f75c91aa963c4312f3
State: online
Cluster Id: 7c1cf54ff4b5ab41f823ac592ba68ca5
Zone: 1
Management Hostnames: node3.example.com
Storage Hostnames: 192.168.1.53
Devices:
 Id:5917085ef4a7beca4f7c61138d152460   Name:/dev/sdb            State:online
            Size (GiB):500     Used (GiB):0       Free (GiB):500
  Bricks:

Node Id: 7c899bc9f50e46efc993dc22263549e4
State: online
Cluster Id: 7c1cf54ff4b5ab41f823ac592ba68ca5
Zone: 1
Management Hostnames: node2.example.com
Storage Hostnames: 192.168.1.52
Devices:
 Id:855490c8fb09e4f21caae9f421f692b0   Name:/dev/sdb            State:online
            Size (GiB):500     Used (GiB):0       Free (GiB):500
  Bricks:

Node Id: c35ba48b042555633b511f459f5aa157
State: online
Cluster Id: 7c1cf54ff4b5ab41f823ac592ba68ca5
Zone: 1
Management Hostnames: node1.example.com
Storage Hostnames: 192.168.1.51
Devices:
 Id:fbf747dc6ccf811fce0196d8280a32e3   Name:/dev/sdb            State:online
            Size (GiB):500     Used (GiB):0       Free (GiB):500
  Bricks:
```

5.3.2 Creating a Volume

This section discusses creating a volume using the Heketi CLI.

To create a volume:

1. Use the `heketi-cli volume create` command to create a volume. This command creates a replicated volume (one brick over three nodes) using a replica count of 3. The size of the volume is 10Gb.

```
# heketi-cli volume create --size=10 --replica=3
Name: vol_2ab33ebc348c2c6dcc3819b2691d0267
Size: 10
Volume Id: 2ab33ebc348c2c6dcc3819b2691d0267
Cluster Id: 7c1cf54ff4b5ab41f823ac592ba68ca5
Mount: 192.168.1.51:vol_2ab33ebc348c2c6dcc3819b2691d0267
Mount Options: backup-volfile-servers=192.168.1.52,192.168.1.53
Block: false
Free Size: 0
Reserved Size: 0
Block Hosting Restriction: (none)
Block Volumes: []
Durability Type: replicate
Distributed+Replica: 3
```

2. Use the `heketi-cli volume list` command to get a list of the volumes managed by Heketi:

```
# heketi-cli volume list
Id:2ab33ebc348c2c6dcc3819b2691d0267    Cluster:7c1cf54ff4b5ab41f823ac592ba68ca5
   Name:vol_2ab33ebc348c2c6dcc3819b2691d0267
```

3. Use the `heketi-cli volume info` command to get information about the volume using the volume `Id`:

```
# heketi-cli volume info 2ab33ebc348c2c6dcc3819b2691d0267
Name: vol_2ab33ebc348c2c6dcc3819b2691d0267
Size: 10
Volume Id: 2ab33ebc348c2c6dcc3819b2691d0267
Cluster Id: 7c1cf54ff4b5ab41f823ac592ba68ca5
Mount: 192.168.1.51:vol_2ab33ebc348c2c6dcc3819b2691d0267
Mount Options: backup-volfile-servers=192.168.1.52,192.168.1.53
Block: false
Free Size: 0
Reserved Size: 0
Block Hosting Restriction: (none)
Block Volumes: []
Durability Type: replicate
Distributed+Replica: 3
```

5.3.3 Expanding a Volume

This section discusses extending a volume using the Heketi CLI.

To extend a volume:

1. Use the `heketi-cli volume expand` command to expand the size of a volume. The volume size in this example adds 10Gb to the volume size.

```
# heketi-cli volume expand --volume=2ab33ebc348c2c6dcc3819b2691d0267 --expand-size=10
Name: vol_2ab33ebc348c2c6dcc3819b2691d0267
Size: 20
Volume Id: 2ab33ebc348c2c6dcc3819b2691d0267
Cluster Id: 7c1cf54ff4b5ab41f823ac592ba68ca5
Mount: 192.168.1.51:vol_2ab33ebc348c2c6dcc3819b2691d0267
Mount Options: backup-volfile-servers=192.168.1.52,192.168.1.53
Block: false
Free Size: 0
Reserved Size: 0
Block Hosting Restriction: (none)
Block Volumes: []
Durability Type: replicate
Distributed+Replica: 3
```

5.3.4 Deleting a Volume

This section discusses deleting a volume using the Heketi CLI.

To delete a volume:

1. Use the `heketi-cli volume list` command to get a list of the volumes managed by Heketi:

```
# heketi-cli volume list
Id:2ab33ebc348c2c6dcc3819b2691d0267    Cluster:7c1cf54ff4b5ab41f823ac592ba68ca5
  Name:vol_2ab33ebc348c2c6dcc3819b2691d0267
```

2. Use the `heketi-cli volume delete` command to delete a volume using the volume `Id`:

```
# heketi-cli volume delete 2ab33ebc348c2c6dcc3819b2691d0267
Volume 2ab33ebc348c2c6dcc3819b2691d0267 deleted
```

5.3.5 Deleting a Device

This section discusses deleting a device using the Heketi CLI. Make sure the device has no volumes listed in the Heketi topology (using the `heketi-cli topology info` command) before you remove it.

To delete a device:

1. Use the `heketi-cli node list` command to get a list of the nodes managed by Heketi:

```
# heketi-cli node list
Id:32755ad123c325f75c91aa963c4312f3 Cluster:7c1cf54ff4b5ab41f823ac592ba68ca5
Id:7c899bc9f50e46efc993dc22263549e4 Cluster:7c1cf54ff4b5ab41f823ac592ba68ca5
Id:c35ba48b042555633b511f459f5aa157 Cluster:7c1cf54ff4b5ab41f823ac592ba68ca5
```

2. Use the `heketi-cli node info` command to get information about the devices on a node using the node `Id`:

```
# heketi-cli node info c35ba48b042555633b511f459f5aa157
Node Id: c35ba48b042555633b511f459f5aa157
State: online
Cluster Id: 7c1cf54ff4b5ab41f823ac592ba68ca5
Zone: 1
Management Hostname: node3.example.com
Storage Hostname: 192.168.1.53
Devices:
Id:fbf747dc6ccf811fce0196d8280a32e3   Name:/dev/sdb  State:online   Size (GiB):500
   Used (GiB):20  Free (GiB):478     Bricks:3
```

3. Use the `heketi-cli device disable` command to disable (take offline) the device using the device `Id`:

```
# heketi-cli device disable fbf747dc6ccf811fce0196d8280a32e3
Device fbf747dc6ccf811fce0196d8280a32e3 is now offline
```

4. Use the `heketi-cli device remove` command to remove the device using the device `Id`:

```
# heketi-cli device remove fbf747dc6ccf811fce0196d8280a32e3
Device fbf747dc6ccf811fce0196d8280a32e3 is now removed
```

5. Use the `heketi-cli device delete` command to delete the device using the device `Id`:

```
# heketi-cli device delete fbf747dc6ccf811fce0196d8280a32e3
Device fbf747dc6ccf811fce0196d8280a32e3 deleted
```

5.3.6 Deleting a Node

This section discusses deleting a node using the Heketi CLI. Make sure the node has no volumes or devices listed in the Heketi topology (using the `heketi-cli topology info` command) before you remove it.

To delete a node:

1. Use the `heketi-cli node list` command to get a list of the nodes managed by Heketi:

```
# heketi-cli node list
Id:32755ad123c325f75c91aa963c4312f3 Cluster:7c1cf54ff4b5ab41f823ac592ba68ca5
Id:7c899bc9f50e46efc993dc22263549e4 Cluster:7c1cf54ff4b5ab41f823ac592ba68ca5
Id:c35ba48b042555633b511f459f5aa157 Cluster:7c1cf54ff4b5ab41f823ac592ba68ca5
```

2. Use the `heketi-cli node disable` command to disable the device using the node `Id`:

```
# heketi-cli node disable c35ba48b042555633b511f459f5aa157
```

```
Node c35ba48b042555633b511f459f5aa157 is now offline
```

3. Use the `heketi-cli node remove` command to remove the node using the node `Id`:

```
# heketi-cli node remove c35ba48b042555633b511f459f5aa157
Node c35ba48b042555633b511f459f5aa157 is now removed
```

4. Use the `heketi-cli node delete` command to delete the node using the node `Id`:

```
# heketi-cli node delete c35ba48b042555633b511f459f5aa157
Node c35ba48b042555633b511f459f5aa157 deleted
```

5.3.7 Deleting a Cluster

This section discusses deleting a cluster using the Heketi CLI. Make sure the cluster has no volumes, nodes or devices listed in the Heketi topology (using the `heketi-cli topology info` command) before you remove it.

To delete a cluster:

1. Use the `heketi-cli cluster list` command to get a list of the clusters managed by Heketi:

```
# heketi-cli cluster list
Clusters:
Id:7c1cf54ff4b5ab41f823ac592ba68ca5 [file][block]
```

2. Use the `heketi-cli cluster delete` command to delete the cluster using the cluster `Id`:

```
# heketi-cli cluster delete 7c1cf54ff4b5ab41f823ac592ba68ca5
Cluster 7c1cf54ff4b5ab41f823ac592ba68ca5 deleted
```

5.3.8 Cleaning up the Heketi Topology

This section shows you how to clean the Heketi topology using the Heketi CLI. You can see the Heketi topology using the `heketi-cli topology info` command.

To clean the Heketi topology:

1. Delete all the volumes. See Section 5.3.4, “Deleting a Volume”.
2. Delete all the devices on each node. See Section 5.3.5, “Deleting a Device”.
3. Delete all the nodes in each cluster. See Section 5.3.6, “Deleting a Node”.
4. Delete all the clusters. See Section 5.3.7, “Deleting a Cluster”.

Chapter 6 Known Issues

The following sections describe known issues in this release.

6.1 Samba Access Fails with SELinux Enabled

If a node in the trusted storage pool on which you want to enable Samba access has SELinux enabled, Samba clients fail to connect to the volume, using both SMB and CIFS.

Connecting to a volume using SMB fails, for example:

```
# smbclient -U user%password //node1/gluster-gv1
tree connect failed: NT_STATUS_UNSUCCESSFUL
```

Connecting to a volume using CIFS also fails, for example:

```
# mount -t cifs -o guest //node1/gluster-gv1 /mnt/glusterfs/
mount error(5): Input/output error Refer to the mount.cifs(8) manual page (e.g. man mount.cifs)
```

To workaround this, you must do one of the following on the node on which you want to enable Samba access:

- Set the SELinux `samba_load_libgfapi` option to `1`.

  ```
  # setsebool -P samba_load_libgfapi 1
  ```

- Disable enforcing mode on SELinux:

  ```
  # setenforce 0
  ```

  To make this change permanent, edit the `/etc/selinux/config` file and change the `SELINUX` variable to `permissive` mode.

(Bug 28701091)

6.2 Samba Access Fails on aarch64 Hardware

Samba clients fail to connect to a Gluster volume, using both SMB and CIFS. This is due to the unavailability of the `samba-vfs-glusterfs` package on the aarch64 platform.

(Bug 29048629)

Gluster Terminology

Brick
: A basic unit of storage in the Gluster file system. A brick is disk storage made available using an exported directory on a server in a trusted storage pool.

Distributed File System
: A file system that allows multiple clients to concurrently access data spread across bricks in a trusted storage pool.

Extended Attributes
: Extended file attributes (abbreviated as xattr) is a file system feature that enables users or programs to associate files and directories with metadata. Gluster stores metadata in xattrs.

Gluster Client
: A Gluster client runs the `glusterfs-client` software to mount gluster storage, either locally or remotely using the Filesystem in Userspace (FUSE) software interface.

Gluster Server
: A Gluster server runs the `glusterd` service daemon to become a node in the trusted storage pool.

Gluster Volume
: A Gluster volume is a logical collection of bricks. Volumes can be distributed, replicated, distributed replicated, dispersed, or distributed dispersed.

Heketi
: A service that exposes an API for scripts to manage trusted storage pools and volumes automatically.

Heketi Cluster
: A Gluster trusted storage pool.

Node
: A host system that is a member of a trusted storage pool.

Split-brain
: A situation where data on two or more bricks in a replicated volume diverges (the content or metadata differs).

Trusted Storage Pool
: A trusted storage pool is a collection of nodes that form a cluster.