Oracle® Linux

Gluster Storage for Oracle Linux User's Guide

F23606_14
April 2022

Oracle Legal Notices

Table of Contents

Preface v
1 Introduction to Gluster Storage for Oracle Linux 1
1.1 About Gluster Storage for Oracle Linux 1
1.2 Notable Updates and New Features 1
1.2.1 Notable Updates and New Features in Release 8 1
1.2.2 Notable Updates and New Features in Release 6 1
1.2.3 Notable Updates and New Features in Release 5 2
1.2.4 Notable Updates and New Features in Release 4.1 2
1.3 Technical Preview Features 3
1.3.1 Technical Preview Features in Release 8 3
1.3.2 Technical Preview Features in Release 6 3
1.3.3 Technical Preview Features in Release 5 4
1.3.4 Technical Preview Features in Release 4.1 4
2 Installing Gluster Storage for Oracle Linux 5
2.1 Hardware and Network Requirements 5
2.2 Operating System Requirements 5
2.3 Enabling Access to the Gluster Storage for Oracle Linux Packages 6
2.3.1 Enabling Access on Oracle Linux 8 Systems 6
2.3.2 Enabling Access on Oracle Linux 7 Systems 8
2.4 Installing and Configuring Gluster 9
2.4.1 Preparing Oracle Linux Nodes 9
2.4.2 Installing the Gluster Server 10
2.4.3 Creating the Trusted Storage Pool 10
2.4.4 Setting Up Transport Layer Security 11
3 Upgrading Gluster Storage for Oracle Linux 13
3.1 Performing an Online Upgrade 13
3.2 Performing an Offline Upgrade 15
3.3 Post Upgrade Requirements 15
3.4 Upgrading Gluster Clients 15
4 Creating and Managing Volumes 17
4.1 Creating Volumes 17
4.1.1 Creating Distributed Volumes 18
4.1.2 Creating Replicated Volumes 18
4.1.3 Creating Distributed Replicated Volumes 19
4.1.4 Creating Dispersed Volumes 21
4.1.5 Creating Distributed Dispersed Volumes 22
4.2 Managing Volumes 23
4.2.1 Setting Volume Options 23
4.2.2 Starting a Volume 23
4.2.3 Stopping a Volume 24
4.2.4 Self Healing a Replicated Volume 24
4.2.5 Expanding a Volume 24
4.2.6 Shrinking a Volume 27
4.2.7 Deleting a Volume 30
4.3 Monitoring Volumes 30
4.3.1 Using the Volume Status Command 30
4.3.2 Using the Volume Profile Command 32
4.3.3 Using the Volume Top Command 34
5 Accessing Volumes 37
5.1 Accessing Volumes by Using iSCSI 37
5.1.1 Installing iSCSI Services 37

5.1.2 Creating a Block Device 37
5.1.3 Accessing an iSCSI Block Device 38
5.2 Accessing Volumes by Using NFS 39
5.3 Accessing Volumes by Using the Gluster Native Client (FUSE) 40
5.4 Accessing Volumes by Using Samba 42
5.4.1 Setting Up the Volume for Samba Access 42
5.4.2 Testing SMB Access to a Volume 43
5.4.3 Testing CIFS Access to a Volume 44
5.4.4 Accessing the Volume from Microsoft Windows 45
6 Automating Volume Lifecycle with Heketi 47
6.1 Installing the Heketi API 47
6.2 Installing the Heketi Client 48
6.3 Using the Heketi CLI 48
6.3.1 Creating a Cluster 49
6.3.2 Creating a Volume 51
6.3.3 Expanding a Volume 52
6.3.4 Deleting a Volume 52
6.3.5 Deleting a Device 53
6.3.6 Deleting a Node 53
6.3.7 Deleting a Cluster 54
6.3.8 Cleaning up the Heketi Topology 54
7 Configuring Geo-replication 55
7.1 About Geo-replication 55
7.2 General Requirements for Geo-Replication 55
7.3 Setting Up Secondary Nodes for Geo-Replication 56
7.4 Configuring the Geo-replication Session 57
7.5 Starting, Stopping and Checking Status of Geo-replication 57
8 Known Issues 59
8.1 Samba Access Fails with SELinux Enabled 59
8.2 Samba Access Fails 59
Gluster Terminology 61

Preface

This document contains information about Release 8 of Gluster Storage for Oracle Linux. 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: 2022-04-04 (revision: 13037)

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:

https://docs.oracle.com/en/operating-systems/oracle-linux/

Conventions

The following text conventions are used in this document:

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

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at https://www.oracle.com/corporate/accessibility/.

For information about the accessibility of the Oracle Help Center, see the Oracle Accessibility Conformance Report at https://www.oracle.com/corporate/accessibility/templates/t2-11535.html.

Access to Oracle Support for Accessibility

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit https://www.oracle.com/corporate/accessibility/learning-support.html#support-tab.

Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive

terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

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.

Note

In this document, the terms *master* and *slave* that were previously used to describe redundant storage architecture have been replaced by the less divisive terms *primary* and *secondary*.

1.1 About Gluster Storage for Oracle Linux

Release 8 of Gluster Storage for Oracle Linux is based on the stable release of the upstream Gluster 8 release and is available for both Oracle Linux 7 and Oracle Linux 8.

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

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 8

Some highlighted features in GlusterFS 8 are:

- Support for healing data in 1 Mb blocks for better performance.
- IPv6 packets are now supported by the `glustereventsd` daemon.
- `fsync` in the replication module uses eager-lock functionality which results in a more efficient performance of VM workloads.

For a complete list of major changes and features that are in GlusterFS 8, see the release notes for GlusterFS 8.0 at https://docs.gluster.org/en/latest/release-notes/8.0/. Subsequent update versions of GlusterFS 8.0 contain bug fixes which are listed in their respective GlusterFS 8.x release notes.

Likewise, the major changes and features that were released in GlusterFS 7 are listed in the release notes for GlusterFS 7.0 at https://docs.gluster.org/en/latest/release-notes/7.0/. Subsequent update versions of GlusterFS 7.0 contain bug fixes which are listed in their respective GlusterFS 7.x release notes.

- **Upgrade.** Upgrade from Gluster Storage for Oracle Linux Release 6.x and 5.x. For reference, see https://docs.gluster.org/en/latest/Upgrade-Guide/upgrade-to-8/

1.2.2 Notable Updates and New Features in Release 6

New features and bug fixes in the upstream release of Gluster between Release 5 and 6 are available in the upstream documentation at:

https://docs.gluster.org/en/latest/release-notes/6.0/

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

- **Introduction of support on Oracle Linux 8 with RHCK.** This release introduces support for Gluster on Oracle Linux 8 when using the Red Hat Compatible Kernel (RHCK).
- **Support for Geo-replication.** The geo-replication feature provides a facility to mirror data across geographically distributed clusters, similar to replicated volumes, but with the primary distinction being the focus on providing asynchronous replication for the purpose of disaster recovery. This feature can be used across a LAN, WAN or across the Internet. Since replication is asynchronous, the architecture follows a primary-secondary (previously, master-slave) model, where changes on the primary volume are replicated to a secondary volume. In the event of a disaster, data can be restored from a secondary volume.
- **Change to process to enable SMB or CIFS exports of Volumes.** It is important to note that in previous versions all volumes were being exported by default via smb.conf in a Samba-CTDB setup. This had some negative implications in terms of performance and the unnecessary exposure of the CTDB lock volume. This release requires that volumes must be explicitly enabled for SMB or CIFS export before the volume is exported via Samba. The configuration remains the same, but an additional step to enable `user.smb` or `user.cifs` on the volume is required.
- **Upgrade.** Upgrade from Gluster Storage for Oracle Linux Release 5, 4.1 and 3.12.

1.2.3 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 Release 5:

- **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 5.1, “Accessing Volumes by 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 6, *Automating Volume Lifecycle with Heketi*, and the upstream documentation at https://github.com/heketi/heketi.
- **Upgrade.** Upgrade from Gluster Storage for Oracle Linux Release 6 and Release 3.12.

1.2.4 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/4.1.0/

In addition to the upstream changes, the following notable features are included in Release 4.1:

- **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 8

The following items are highlighted as technical preview features in Release 8 of Gluster Storage for Oracle Linux. For reference, see https://docs.oracle.com/en/operating-systems/oracle-linux/gluster-storage/gluster-intro.html#gluster-tech-preview-8.

- **Ansible Modules.** Ansible modules for Gluster are made available in this release for use within the context of Oracle Linux Virtualization Manager. These modules are available as a technical preview for use outside of Oracle Linux Virtualization Manager and remain unsupported when used in this way.

  The Gluster Ansible module RPMs are included for this purpose:

  - `gluster-ansible-cluster`
  - `gluster-ansible-features`
  - `gluster-ansible-infra`
  - `gluster-ansible-maintenance`
  - `gluster-ansible-repositories`
  - `gluster-ansible-roles`

  The following RPMs are included as dependencies and are also unsupported outside of the context of Oracle Linux Virtualization Manager:

  - `ansible`
  - `sshpass`
  - `python-httplib2`
  - `python2-jemspath`

1.3.2 Technical Preview Features in Release 6

The following items are highlighted as technical preview features in Release 6 of Gluster Storage for Oracle Linux:

- **Ansible Modules.** Ansible modules for Gluster are made available in this release for use within the context of Oracle Linux Virtualization Manager. These modules are available as a technical preview for use outside of Oracle Linux Virtualization Manager and remain unsupported when used in this way.

  The Gluster Ansible module RPMs are included for this purpose:

- `gluster-ansible-cluster`
- `gluster-ansible-features`
- `gluster-ansible-infra`
- `gluster-ansible-maintenance`
- `gluster-ansible-repositories`
- `gluster-ansible-roles`

The following RPMs are included as dependencies and are also unsupported outside of the context of Oracle Linux Virtualization Manager:

- `ansible`
- `sshpass`
- `python-httplib2`
- `python2-jemspath`

- **GlusterD-2.0.** GlusterD-2.0 (`glusterd2`), the thin management layer for Gluster with container orchestration systems, remains a technical preview in Release 6. GlusterD-2.0 is a re-implementation of `glusterd`. `glusterd2` purports to have better consistency, scalability and performance when compared with the current `glusterd`, while also being more modular, 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.

1.3.3 Technical Preview Features in Release 5

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

- **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.

1.3.4 Technical Preview Features in Release 4.1

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

- **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 consists of the following:

- 2 CPU cores
- 2GB RAM
- 1GB Ethernet NIC

  Recommended: 2 x 1GB Ethernet NICs in a bonded (802.3ad/LACP) configuration
- Dedicated storage sized for your data requirements (10GB or higher) and formatted as an XFS file system

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

Release 8 of Gluster Storage for Oracle Linux 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 8	Oracle Linux 8.2	Unbreakable Enterprise Kernel Release 6 (UEK R6)

Platform	Operating System Release	Minimum Operating System Maintenance Release	Kernel
			Red Hat Compatible Kernel (RHCK)
aarch64	Oracle Linux 8	Oracle Linux 8.2	UEK R6 RHCK
x86_64	Oracle Linux 7	Oracle Linux 7.7	UEK R6 UEK R5 UEK R4 RHCK
aarch64	Oracle Linux 7	Oracle Linux 7.7	UEK R6 UEK R5

Important

- All nodes within your deployment must run the same Oracle Linux release and update level. Otherwise, changes between releases in some component software, such as the default Python version, might cause communication issues between nodes.
- If you are upgrading from a previous release of Gluster Storage for Oracle Linux, you must also upgrade to the latest update level of the Oracle Linux release that you are using.

2.3 Enabling Access to the Gluster Storage for Oracle Linux Packages

Note

When working on Oracle Linux 8 systems, where the documentation uses the `yum` command in the examples, you can substitute the command with `dnf` for the same behavior.

2.3.1 Enabling Access on Oracle Linux 8 Systems

The Gluster Storage for Oracle Linux packages are available on the Oracle Linux yum server in the `ol8_gluster_appstream` repository, or on the Unbreakable Linux Network (ULN) in the `ol8_`*`arch`*`_gluster_appstream` channel.

Enabling Repositories with ULN

If you are registered to use ULN, use the ULN web interface to subscribe the system to the appropriate 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 to ensure that all dependencies are met:
   - `ol8_arch_gluster_appstream`
   - `ol8_arch_baseos_latest`
   - `ol8_arch_appstream`
5. Click **Save Subscriptions**.

Enabling Repositories with the Oracle Linux Yum Server

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

1. Install the `oracle-gluster-release-el8` release package to install the Gluster Storage for Oracle Linux yum repository configuration.

   ```
   sudo dnf install oracle-gluster-release-el8
   ```

2. Enable the following yum repositories:
   - `ol8_gluster_appstream`
   - `ol8_baseos_latest`
   - `ol8_appstream`

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

   ```
   sudo dnf config-manager --enable ol8_gluster_appstream ol8_baseos_latest ol8_appstream
   ```

Oracle Linux 8: Enabling the glusterfs Module and Application Stream

On Oracle Linux 8, the Gluster Storage for Oracle Linux packages are released as an application stream module. All of the packages specific to a particular release of the Gluster software are released within a stream. Furthermore, packages are bundled within a profile to be installed as a single step on a system depending on a use case.

Two modules are available for Gluster. The `glusterfs` module contains all supported packages required to install and run a Gluster server node or a Gluster client. The `glusterfs-developer` module contains packages that are released as a technical preview for developer use only. Packages that are available in the `glusterfs-developer` module are unsupported or may only be supported within very specific contexts, such as, when they are made available for other supported software.

To gain access to the Gluster Storage for Oracle Linux packages, enable the `glusterfs` application stream module:

```
sudo dnf module enable glusterfs
```

Once the module is enabled, you can install any of the packages within the module by following any of the other instructions in this documentation.

You are also able to take advantage of the module profiles to simply install everything you might need for a use case. For example, on any server node, you can run:

```
sudo dnf install @glusterfs/server
```

This action installs all of the possible packages required for any Gluster server node, including core Gluster functionality, Gluster geo-replication functionality, NFS-Ganesha server and Heketi server packages.

On a client system where you intend to access or mount a Gluster share, you can run:

```
sudo dnf install @glusterfs/client
```

This action installs the Gluster packages required to mount a Gluster share or to act as a Heketi client.

2.3.2 Enabling Access on Oracle Linux 7 Systems

The Gluster Storage for Oracle Linux packages are available on the Oracle Linux yum server in the `ol7_gluster8` repository, or on the Unbreakable Linux Network (ULN) in the `ol7_arch_gluster8` 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.

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_arch_gluster8`
   - `ol7_arch_addons`
   - `ol7_arch_latest`
   - `ol7_arch_optional_latest`
   - `ol7_arch_UEKR5` or `ol7_arch_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, use the command line to enable the Gluster Storage for Oracle Linux yum repository.

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

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

2. Enable the following yum repositories:
   - `ol7_gluster8`
   - `ol7_addons`
   - `ol7_latest`

- `ol7_optional_latest`
- `ol7_UEKR5` or `ol7_UEKR4`

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

```
sudo yum-config-manager --enable ol7_gluster8 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*.

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

2.4.1 Preparing Oracle Linux Nodes

In addition to the requirements listed in Section 2.1, “Hardware and Network Requirements” and Section 2.2, “Operating System Requirements”, all the Oracle Linux systems that you intend to use as nodes must have the following configurations and features:

- The same storage configuration
- Synchronized time
- Resolvable fully qualified domain names (FQDNs)

Storage Configuration

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:

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

Synchronized Time

Time must be accurate and synchronized across the nodes in the pool. For this purpose, you can install and configure NTP or PTP on each node. For more information on configuring synchornizing time on Oracle Linux 7, see *Configuring Network Time* in *Oracle® Linux 7: Setting Up Networking*.

For equivalent information that applies to Oracle Linux 8, see *Oracle Linux: Update the system date and time from the command line interface* and *Configuring Network Time* in *Oracle® Linux 8: Setting Up Networking*.

Resolvable Host Names

All nodes must be able to resolve the FQDN for each node within the pool. You can either use DNS or provide entries within `/etc/hosts` for each system. If you rely on DNS, the service 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
```

2.4.2 Installing the Gluster Server

Install the Gluster server packages on each node to be included in a trusted storage pool.

1. Install the `glusterfs-server` package.

   If running Oracle Linux 8, type the following command:

   ```
   sudo dnf install @glusterfs/server
   ```

   Otherwise, type:

   ```
   sudo yum install glusterfs-server
   ```

2. Start and enable the Gluster server service:

   ```
   sudo systemctl enable --now glusterd
   ```

3. Adjust firewall configuration.

   Pool network communications must be able to take place between nodes within the cluster. If you want to deploy a firewall on any of the nodes, then configure the service 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:

     ```
     sudo firewall-cmd --permanent --change-zone=if-name --zone=trusted
     sudo firewall-cmd --reload
     ```

     The commands automatically add the line `zone=trusted` to an interface's `/etc/sysconfig/network-scripts/ifcfg-if-name` file and then reload the firewall to activate the change.

     With this configuration, your clients must be on the same dedicated network and configured for the same firewall zone. If necessary, you can also configure additional rules specific to the interface on which your clients are connecting if needed to further customize the firewall configuration.

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

     ```
     sudo firewall-cmd --permanent --add-service=glusterfs
     sudo 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.

2.4.3 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.

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:

```
sudo gluster peer probe node2
sudo gluster peer probe node3
```

2. (Optional) Check the status of each node in the pool.

```
sudo gluster peer status
```

3. (Optional) List the nodes in the pool.

```
sudo gluster pool list
```

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

```
sudo gluster peer detach hostname
```

2.4.4 Setting Up Transport Layer Security

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

Gluster performs mutual authentication in all transactions. If one side of a connection is configured to use TLS, then the other side must use TLS as well. Every node must have a copy either of the public certificate of every other node in the pool, or of the signing CA certificate that can be used 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. In turn, 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, you should use certificates that are properly signed by a Certificate Authority (CA) to improve validation security and also reduces the complexity of configuration. However, in cases where numerous clients access the pool, this method is not always practical. This section describes configuration for environments where certificates are signed by a CA and when certificates are self-signed.

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

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

2. Perform one of the following substeps depending on the method you adopt.

   Using self-signed certificates

   a. Create a self-signed certificate on each node in the storage pool.

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

   b. Concatenate the contents of each of `*.pem` files into a single file called `/etc/ssl/glusterfs.ca`.

   c. Ensure that the `/etc/ssl/glusterfs.ca` exists on every node in the pool.

Each node uses this file to validate the certificates presented by other nodes or clients that connect to the node. 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.

Using CA-signed certificates

a. Create a certificate signing request. (CSR)

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

b. After obtaining 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 by creating an empty file at `/var/lib/glusterd/secure-access` on each node in the pool.

   Perform this same step on any client system where you intend to mount a volume.

```
sudo 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`, type the following:

```
sudo gluster volume set myvolume client.ssl on
sudo 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.

```
sudo systemctl restart glusterd
```

Chapter 3 Upgrading Gluster Storage for Oracle Linux

This section discusses upgrading to Release 8 of Gluster Storage for Oracle Linux from previous releases such as Releases 6, 5, 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 6.** `ol7_gluster6` repository or `ol7_arch_gluster6` ULN channel.
- **Release 5.** `ol7_gluster5` repository or `ol7_arch_gluster5` ULN channel.
- **Release 4.1.** `ol7_gluster41` repository or `ol7_arch_gluster41` ULN channel.
- **Release 3.12.** `ol7_gluster312` repository or `ol7_arch_gluster312` ULN channel.

Important

Do not make any configuration changes during the upgrade. Upgrade the servers first before upgrading clients. After the upgrade, ensure that your entire Gluster deployment is running the same Gluster server and client versions.

3.1 Performing an Online Upgrade

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

Only replicated and distributed replicated volumes can be upgraded online. Any other volume types must be upgraded offline. See Section 3.2, “Performing an Offline Upgrade” for information on performing an offline upgrade.

Perform the following procedure on each Gluster node. The procedure assumes that multiple replicas of a replica set are not part of the same server in the trusted storage pool.

1. Ensure that your system is running the latest update level of the operating system.

   If not, then run the following commands:

   ```
   sudo yum update
   sudo reboot
   ```

2. Update to the new directory structure introduced in Release 8 of Gluster Storage for Oracle Linux.

   The new directory structure specifically serves changelog files that are related to geo-replication. The script moves these files to the new required location.

   a. Download the `glusterfs-georep-upgrade.py` script by using one of the following links:

      - https://raw.githubusercontent.com/gluster/glusterfs/devel/extras/glusterfs-georep-upgrade.py
      - https://github.com/gluster/glusterfs/blob/devel/extras/glusterfs-georep-upgrade.py

   b. Stop the geo-replication session.

```
sudo gluster volume geo-replication primary_volume georep@secondary-node1.example.com::secondaryvol stop
```

c. Use the script to migrate the georeplication `changelogs` to the new structure.

```
sudo glusterfs-georep-upgrade.py path-to-brick
```

3. If you are upgrading from Gluster Storage for Oracle Linux Release 3.12, you need to unset some deprecated features. Do the following:

   a. Check if the `lock-heal` or `grace-timeout` features are being used.

   ```
   sudo gluster volume info
   ```

   b. If these parameters are listed under the `Options Reconfigured` section of the output, unset these parameters with the following commands:

   ```
   sudo gluster volume reset myvolume features.lock-heal
   sudo gluster volume reset myvolume features.grace-timeout
   ```

4. Stop the Gluster service and all Gluster file system processes, and then verify that no related processes are running.

   ```
   sudo systemctl stop glusterd
   sudo killall glusterfs glusterfsd
   sudo ps aux | grep gluster
   ```

5. Stop any Gluster related services, such as Samba and NFS-Ganesha.

   ```
   sudo systemctl stop smb
   sudo systemctl stop nfs-ganesha
   ```

6. Update the Gluster Storage for Oracle Linux packages:

   ```
   sudo yum update glusterfs-server
   ```

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

   ```
   sudo yum update nfs-ganesha-gluster
   ```

8. Start the Gluster service and any Gluster related services that you might be using, for example, Samba and NFS-Ganesha.

   ```
   sudo systemctl daemon-reload
   sudo systemctl start glusterd
   sudo systemctl start smb
   sudo systemctl start nfs-ganesha
   ```

9. Check for bricks that might be offline and bring them online.

   ```
   sudo gluster volume status
   sudo gluster volume start volume_name force
   ```

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

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

11. When all bricks are online, heal the volumes, then optionally view the healing information for each volume.

    ```
    sudo for i in `gluster volume list`; do gluster volume heal $i; done
    sudo gluster volume heal volume_name info
    ```

12. If a deployed geo-replication session was stopped for the upgrade, then restart the session.

```
sudo gluster volume geo-replication primary_volume georep@secondary-node1.example.com::secondaryvol sta
```

3.2 Performing 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.

1. Stop the volume.

```
sudo gluster volume stop myvolume
```

2. Upgrade all Gluster nodes using the steps provided in Section 3.1, “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. After the upgrade, restart the volume.

```
sudo gluster volume start myvolume
```

3.3 Post Upgrade Requirements

Complete this procedure after either an online or an offline upgrade.

1. Check the operating version number for all volumes.

```
sudo gluster volume get all cluster.op-version
Option                                  Value
------                                  -----
cluster.op-version                      60000
```

2. If the parameter value is not 6000, set the version number accordingly.

```
sudo gluster volume set all cluster.op-version 60000
```

3. Upgrade the clients that access the volumes. See Section 3.4, “Upgrading Gluster Clients” for information on upgrading Gluster clients.

3.4 Upgrading Gluster Clients

When the Gluster server nodes have been upgraded, you should upgrade Gluster clients. Complete this procedure on every client.

1. Unmount all Gluster mount points on the client.
2. Stop all applications that access the volumes.
3. For Gluster native clients (FUSE), update Gluster.

```
sudo yum update glusterfs glusterfs-fuse
```

4. Mount all Gluster shares.
5. Start any applications that were stopped for the upgrade.

Chapter 4 Creating and Managing Volumes

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

4.1 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 5, *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 2.4.4, “Setting Up Transport Layer Security” for more information.

For more detailed information, see the Gluster upstream documentation.

4.1.1 Creating Distributed Volumes

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

Example 4.1 Creating a distributed volume

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

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

4.1.2 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 4.2 Creating a replicated volume

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

```
sudo 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
sudo gluster volume start myvolume
volume start: myvolume: success
sudo 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 4.3 Creating a replicated volume with an arbiter

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

```
sudo 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
sudo gluster volume start myvolume
volume start: myvolume: success
sudo 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
```

4.1.3 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 4.4 Creating a distributed replicated volume with one brick on six nodes

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

```
sudo 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
sudo gluster volume start myvolume
volume start: myvolume: success
sudo 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 4.5 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.

```
sudo 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
sudo gluster volume start myvolume
volume start: myvolume: success
sudo 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 4.6 Creating a distributed replicated volume with two bricks over three nodes

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

```
sudo 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
sudo gluster volume start myvolume
volume start: myvolume: success
sudo 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 4.7 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.

```
sudo 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
sudo gluster volume start myvolume
volume start: myvolume: success
sudo 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
```

4.1.4 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 4.8 Creating a dispersed volume with one brick on three nodes

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

```
sudo 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
sudo gluster volume start myvolume
volume start: myvolume: success
sudo 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
```

4.1.5 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 4.9 Creating a distributed dispersed volume with one brick on six nodes

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

```
sudo 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
sudo gluster volume start myvolume
volume start: myvolume: success
sudo 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 4.10 Creating a distributed dispersed volume with two bricks on three nodes

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

```
sudo 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
sudo gluster volume start myvolume
volume start: myvolume: success
sudo 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
```

4.2 Managing Volumes

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

4.2.1 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:

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

Likewise, to set access to volume subdirectories, type:

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

4.2.2 Starting a Volume

To start a volume, use the the command:

`gluster volume start volume_name`

4.2.3 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.

4.2.4 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:

```
sudo gluster volume heal myvolume info
```

You can start a self-healing manually using:

```
sudo gluster volume heal myvolume
```

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

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

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

4.2.5 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.

   ```
   gluster peer probe node4
   ```
3. Add the brick(s).

   ```
   gluster volume add-brick myvolume node4:/data/glusterfs/myvolume/mybrick/brick
   ```
4. Rebalance the volume to distribute files to the new brick(s).

   ```
   sudo gluster volume rebalance myvolume start
   ```

   To check the status of the volume rebalance, type:

   ```
   sudo gluster volume rebalance myvolume status
   ```

Example 4.11 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.

```
sudo 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
sudo gluster volume start myvolume
volume start: myvolume: success
sudo 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

sudo gluster peer status
Number of Peers: 2

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

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

sudo gluster peer probe node4
peer probe: success.
sudo 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)

sudo gluster volume add-brick myvolume replica 4 \
node4:/data/glusterfs/myvolume/mybrick/brick1 \
node4:/data/glusterfs/myvolume/mybrick/brick2
volume add-brick: success
sudo 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
sudo 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: ...
sudo gluster volume rebalance myvolume status
...
volume rebalance: myvolume: success
```

Example 4.12 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 4.11, “Creating a distributed replicated volume and adding a node”.

```
sudo 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
sudo 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.
sudo 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: ...
sudo gluster volume rebalance myvolume status
...
volume rebalance: myvolume: success
```

4.2.6 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).

   ```
   sudo 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. To check the status of the brick removal, type:

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

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

   ```
   sudo 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 4.13 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.

```
sudo 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
sudo 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
sudo 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
sudo gluster peer detach node4
peer detach: success
sudo gluster peer status
Number of Peers: 2

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

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

Example 4.14 Removing bricks from a distributed replicated volume

This example removes two bricks from a distributed replicated volume.

```
sudo 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
sudo 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: ...
sudo 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

sudo 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.
sudo 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
```

4.2.7 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:

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

4.3 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.

4.3.1 Using the Volume Status Command

The `gluster volume status` command displays information on the status of bricks and volumes. Use the following syntax:

```
gluster volume status volume_name options
```

The following examples show basic use of the `gluster volume status` command that can be used for the performance of common tasks. For more information, see the upstream documentation.

`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 4.15 Showing status information about bricks in a volume

This example displays status information about bricks in a volume.

```
sudo 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 4.16 Showing detailed status information about bricks in a volume

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

```
sudo 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 4.17 Showing information about memory usage for bricks in a volume

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

```
sudo 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

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

4.3.2 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.

Use the following syntax:

`gluster volume profile` *`volume_name options`*

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

The following exmples show basic use of the `gluster volume profile` command and are useful for the performance of common tasks. For more information, see the upstream documentation.

`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 4.18 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`).

```
sudo gluster volume profile myvolume start
Starting volume profile on myvolume has been successful
sudo 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
sudo 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
...
sudo gluster volume profile myvolume stop
```

```
Stopping volume profile on myvolume has been successful
```

4.3.3 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). Use the following syntax:

`gluster volume top` *`volume_name options`*

To display the full syntax of the command, type `gluster volume top -help`.

The following examples show basic use of the `gluster volume top` command and are useful for the performance of common tasks. For more information , see the upstream documentation.

`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 4.19 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.

```
sudo 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
sudo 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 4.20 Showing performance for a brick

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

```
sudo gluster volume top myvolume read-perf bs 2014 count 1024 brick node1:/data/glusterfs/myvolume/mybrick/
Brick: node1:/data/glusterfs/myvolume/mybrick/brick
Throughput 1844.67 MBps time 0.0011 secs
sudo 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 5 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.

5.1 Accessing Volumes by 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.

5.1.1 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.

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

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

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

5.1.2 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.

   ```
   sudo gluster-block create myvolume/myvolume-block ha 3 prealloc no 192.168.1.51,192.168.1.52,192.168.1.
   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.

```
sudo gluster-block list myvolume
myvolume-block
```

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

```
sudo 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.

```
sudo 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]
...
```

5.1.3 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.

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

2. Enable the `iscsid` service:

```
sudo 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:

```
sudo 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:

```
sudo 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.

   a. Load and enable the multipath module.

```
sudo modprobe dm_multipath
sudo mpathconf --enable
```

b. Restart and enable the `multipathd` service.

```
sudo systemctl restart multipathd
sudo systemctl enable multipathd
```

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

```
sudo 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:

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

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

```
sudo mount /dev/mapper/mpatha /mnt/
```

5.2 Accessing Volumes by 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.

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

1. Install the Gluster NFS-Ganesha client packages:

```
sudo 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:

```
sudo 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:

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

4. Check the volume is exported:

```
sudo 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:

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

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

5.3 Accessing Volumes by 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.

Do the following:

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:

   If running Oracle Linux 8, type the following command:

```
sudo dnf install @glusterfs/client
```

   Otherwise, type:

```
sudo yum install glusterfs glusterfs-fuse
```

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

```
sudo 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 2.4.4, “Setting Up Transport Layer Security” 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:

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

   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:

   ```
   sudo mkdir -p /var/lib/glusterfs
   sudo 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:

   ```
   sudo gluster volume stop myvolume
   sudo systemctl restart glusterd
   sudo 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:

   ```
   sudo 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:

   ```
   sudo 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:

   ```
   sudo 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
```

5.4 Accessing Volumes by 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*.

5.4.1 Setting Up the Volume for Samba Access

The following procedure sets 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.

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

1. Install the Samba packages for Gluster:

   ```
   sudo yum install samba samba-vfs-glusterfs
   ```

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

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

3. Enable and start the Samba service:

   ```
   sudo 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:

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

   Restart the Samba service:

   ```
   sudo 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:

```
sudo systemctl restart smb
```

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

```
sudo gluster volume stop myvolume
sudo gluster volume set myvolume user.smb enable
sudo gluster volume start myvolume
```

Note that when setting the SMB or CIFS option on the volume, you can use either `user.smb` or `user.cifs` to enable the type of export you require. Note that if both options are enabled, `user.smb` has precedence.

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, as long as the `user.smb` or `user.cifs` option is set for the volume. 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:

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

To re-enable the hooks, you can run:

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

5.4.2 Testing SMB Access to a Volume

The following procedure 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.

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

```
sudo 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:

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

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

```
sudo 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`.

5.4.3 Testing CIFS Access to a Volume

The following procedure 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:

```
sudo yum install cifs-utils
```

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

```
sudo 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:

```
sudo 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:

```
sudo 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.

```
sudo chmod 600 /credfile
```

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

```
sudo 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:

```
sudo chmod 777 /gluster-storage
```

5.4.4 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 6 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 2.4.3, “Creating the Trusted Storage Pool” and Section 4.1, “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.

6.1 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.

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.

   ```
   sudo 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:

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

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

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

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

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

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

4. Configure other Heketi service options, as required, 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. Review the default settings in the `/etc/heketi/heketi.json` file.

   This file is included by default when you install the `heketi` package. Some of the example entries in the file are invalid variables or instructions, which if not modified, causes a failure. For example, you must either remove or modify optional entries such as the `port` and `fstab` entries to replace the default values.

   Make sure that the configuration entries in this file are properly set to ensure that the Heketi service runs properly.

6. Start and enable the `heketi` service.

   ```
   sudo systemctl enable --now heketi
   ```

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

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

6.2 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.

```
sudo yum install heketi-client
```

6.3 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.

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

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

6.3.1 Creating a Cluster

Follow this procedures to create a cluster with the Heketi CLI.

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:

   ```
   sudo 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.

```
sudo 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.

```
sudo 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.

```
sudo 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:
```

6.3.2 Creating a Volume

Follow this procedure to create a volume by using the Heketi CLI.

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.

```
sudo 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.

```
sudo 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`.

```
sudo 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
```

6.3.3 Expanding a Volume

Follow this procedure to extend a volume by using the Heketi CLI.

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.

```
sudo 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
```

6.3.4 Deleting a Volume

Follow this procedure to delete a volume by 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.

```
sudo 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`.

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

6.3.5 Deleting a Device

Follow this procedure to delete a device by using the Heketi CLI. Ensure that the device has no volumes listed in the Heketi topology (using the `heketi-cli topology info` command) before you remove it.

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

```
sudo 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`.

```
sudo 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`.

```
sudo 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`.

```
sudo 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`.

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

6.3.6 Deleting a Node

Follow this procedure to delete a node by using the Heketi CLI. Run the `heketi-cli topology info` and ensure that the node has no volumes or devices listed in the Heketi topology before you remove the node.

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

```
sudo 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`.

```
sudo 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`.

```
sudo 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`.

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

6.3.7 Deleting a Cluster

Follow this procedure to delete a cluster by using the Heketi CLI. Run the `heketi-cli topology info` command and ensure that the cluster has no volumes, nodes or devices listed in the Heketi topology before you remove the cluster.

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

```
sudo 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`.

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

6.3.8 Cleaning up the Heketi Topology

Follow this procedure to clean the Heketi topology by using the Heketi CLI. Display the Heketi topology by using the `heketi-cli topology info` command.

1. Delete all the volumes. See Section 6.3.4, “Deleting a Volume”.
2. Delete all the devices on each node. See Section 6.3.5, “Deleting a Device”.
3. Delete all the nodes in each cluster. See Section 6.3.6, “Deleting a Node”.
4. Delete all the clusters. See Section 6.3.7, “Deleting a Cluster”.

Chapter 7 Configuring Geo-replication

Geo-replication provides a facility to mirror data across geographically distributed clusters for the purpose of disaster recovery. The mirroring process is asynchronous, in that it is run periodically and only files that have been modified are replicated. This feature can be used across a LAN, WAN or across the Internet. Since replication is asynchronous, the architecture follows a primary-secondary storage model, where changes on the primary volume are replicated to a secondary volume. In the event of a disaster, data can be restored from a secondary volume.

This chapter describes geo-replication processes, the minimum requirements, and the steps, including commands used, to configure your environment to deploy the feature.

7.1 About Geo-replication

Geo-replication's paramount use case is for disaster recovery. It performs asynchronous copies of data from one Gluster volume to another hosted in a separate geographically distinct cluster. Copies are performed by using a backend `rsync` process over an SSH connection.

Geo-replication assumes that a *primary* cluster hosts data in a configured volume. Changes made to data on the primary cluster are copied, periodically, to a volume configured on a *secondary* cluster.

In the event of failure of the primary cluster, data can be restored from the secondary cluster, which can also be promoted to replace the primary cluster.

Gluster is agnostic about the actual physical location or network between separate clusters where geo-replication is configured. You can configure geo-replication on the same LAN, on a WAN, across the Internet, or even between virtual machines hosted on the same system. Provided that the clusters involved can access each other through SSH across a network, geo-replication can be configured.

Geo-replication also facilitates the possibility of multi-site mirroring, where data from a primary cluster is copied to multiple secondary clusters, and also cascading mirroring, where data that is replicated to a secondary cluster can in turn be mirrored to another secondary cluster.

This documentation focuses on a basic, tested configuration where data on a standard Gluster volume is geo-replicated to a volume hosted on a single secondary cluster.

For more information on geo-replication, see the upstream documentation at https://docs.gluster.org/en/latest/Administrator%20Guide/Geo%20Replication/

7.2 General Requirements for Geo-Replication

Geo-replication requires at least two gluster clusters to be configured to the level where a volume is accessible on each cluster. One cluster is treated as the primary cluster, and the second is treated as the secondary cluster. The primary cluster may already be in use prior to the configuration of a secondary cluster.

Clusters must be set up and configured following the requirements set out in Chapter 2, *Installing Gluster Storage for Oracle Linux*. Notably, all nodes in both clusters must have network synchronized time using an NTP service, such as `chrony`.

Geo-replication is performed over SSH between primary and secondary clusters. Access should be available for all nodes in all clusters that take part in geo-replication, so that failover between nodes is handled appropriately. Networking and firewall configuration must facilitate this access.

Ensure that the secondary cluster volume that you are using for mirroring is clean of data and has sufficient capacity to mirror the content on the primary cluster.

Install the `glusterfs-geo-replication` package on *all* nodes in *both* clusters. For example:

```
sudo yum install glusterfs-geo-replication
```

Other general software requirements are met by default on most Oracle Linux systems, but you should check that you have the latest available versions of the following packages:

- `rsync`
- `openssh`
- `glusterfs`
- `glusterfs-fuse`

Important

When using geo-replication in conjunction with the Gluster snapshot functionality, ensure that snapshots are not out of order on both primary and secondary nodes during restoration. Therefore, when performing snapshot operations, always pause geo-replication first. After the session is paused, either take a snapshot of a volume or restore a previous snapshot. These operations must be done on on both primary and secondary nodes. Then, resume the geo-replicatoin session.

7.3 Setting Up Secondary Nodes for Geo-Replication

If the SSH connections that are used to synchornize the data are required to connect to the root user, then the geo-replication service can connect from the primary cluster to the secondary cluster by using that root account. However, exposing ssh access for a root user is discouraged for security reasons. On production systems, you should instead create a user and group specifically for the purpose of handling these connections on each of the secondary node systems.

```
sudo useradd georep
```

The *`georep`* is the user you intend to use for connections between nodes. You can set the variable to an existing group that has been created for this user. Alternatively, you can create a separate group to which you add the new user.

You must set the password for this user on at least one secondary host to enable you to copy an ssh key to the host later during the configuration.

You can now configure the gluster mountbroker to automatically handle mounting the volume that you intend to use for mirroring with the appropriate permissions. For this task, you run the `gluster-mountbroker` on any single secondary node system.

1. Set up the gluster mountbroker for the new account that you have created:

   ```
   sudo gluster-mountbroker setup /var/mountbroker-root georep
   ```

   This command sets up a root folder for all volumes handled by the mountbroker. Typically, the folder is set to `/var/mountbroker-root`, but you can define any location on your secondary nodes. The command creates the directory on all nodes. Substitute *`georep`* with the *group* that has access permissions to this folder. Typically, the *`georep`* matches the the username that you previously created. However, if more than one user can access this data, you might need to define a broader group.

2. Add a volume on your existing secondary cluster to the mountbroker.

```
sudo gluster-mountbroker add secondaryvol georep
```

The *secondaryvol* is the volume that you intend to use on your secondary cluster.

3. Check the status of the mountbroker to determine whether everything is set up correctly:

```
sudo gluster-mountbroker status
```

4. Restart the glusterd service on *all* of the secondary cluster nodes:

```
sudo systemctl restart glusterd
```

7.4 Configuring the Geo-replication Session

This procedure creates the ssh key which is then copied to the authorized_keys file of each node of the secondary cluster.

1. On a node in the primary cluster, create an initial ssh key.

```
sudo ssh-keygen
```

2. At the passphrase prompt, press Enter.

3. Copy the created key to a node in your secondary cluster for the *georep* user.

```
sudo ssh-copy-id georep@secondary-node1.example.com
```

4. On the same primary cluster node where you created the ssh key, run the following command:

```
sudo gluster-georep-sshkey generate
```

The command creates a separate key that is copied to all of the primary nodes. The key is then used for subsequent communications with any secondary nodes during geo-replication.

5. Still on the same node, create the geo-replication session.

```
sudo gluster volume geo-replication myvolume georep@secondary-node1.example.com::secondaryvol create pu
```

myvolume	Volume on the primary cluster to be replicated
secondary-node1.example.com	FQDN or IP address of the node on the secondary cluster.
secondaryvol	Volume on the secondary cluster to be used for replication.

The command copies the public keys for the primary cluster nodes to the node of the secondary cluster.

6. On the secondary node, configure the environment to use these keys for the *georep*.

```
sudo /usr/libexec/glusterfs/set_geo_rep_pem_keys.sh georep myvolume secondaryvol
```

7.5 Starting, Stopping and Checking Status of Geo-replication

After configuring the geo-replication session, start the replication with the following command:

```
sudo gluster volume geo-replication myvolume georep@secondary-node1.example.com::secondaryvol start
```

Replace `start` with `stop` to stop geo-replication for the volume.

To see the status of all running geo-replication, run this command:

```
sudo gluster volume geo-replication status
```

The command output shows which cluster nodes in each cluster are being used for the geo-replication synchronization. The status of a geo-replication session can be any of the following:

- `Initializing`: The session is starting and the first connections are being made.
- `Created`: The geo-replication session is created, but not started.
- `Active`: The `gsync` daemon in this node is active and syncing the data. Only one replica pair on the primary cluster is ever in Active state and handling data synchronization. If the Active node fails, a Passive node is promoted to Active.
- `Passive`: A replica pair of the active node. Passive nodes in the cluster are on standby to be promoted to Active status if the currently Active node fails.
- `Faulty`: Faulty status can indicate a temporary or critical problem in the geo-replication session. In some cases, the Faulty status might autorectify as the geo-replication service attempt to restart on a node. If a Faulty status persists, check log files to try to resolve the issue. You can determine which log file you should look at with the `config` command, for example:

  ```
  sudo gluster volume geo-replication myvolume georep@secondary-node1.example.com::secondaryvol config log-fil
  ```

- `Stopped`: The geo-replication session is stopped.

Chapter 8 Known Issues

The following sections describe known issues in this release.

8.1 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:

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

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

```
sudo 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)
```

As a workaround, set the `samba_load_libgfapi` parameter on the node on which you want to enable Samba access:

```
sudo setsebool -P samba_load_libgfapi 1
```

(Bug 28701091)

8.2 Samba Access Fails

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. This issue is encountered on both x86_64 and aarch64 Oracle Linux 8 systems. On Oracle Linux 7 systems, the issue is encountered only on aarch64 platforms.

As a workaround in the absence of the `samba-vfs-glusterfs` package, you can mount the volume by using FUSE, which is the Gluster native mount, in the same system where the Samba server is running. Then you would share the mount point by using the Samba server.

(Bugs 29048629; 30205755)

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.