Red Hat Gluster Storage 3.4 Administration Guide

Configuring and Managing Red Hat Gluster Storage

Red Hat Gluster Storage Documentation Team

Red Hat Gluster Storage 3.4 Administration Guide

Configuring and Managing Red Hat Gluster Storage

Red Hat Gluster Storage Documentation Team
gluster-docs@redhat.com

Legal Notice

Abstract

Red Hat Gluster Storage Administration Guide describes the configuration and management of Red Hat Gluster Storage for On-Premise.

Table of Contents

Part I. Preface **5**

Chapter 1. Preface **6**
1.1. About Red Hat Gluster Storage 6
1.2. About glusterFS 6
1.3. About On-premises Installation 6

Part II. Overview **7**

Chapter 2. Architecture and Concepts **8**
2.1. Architecture 8
2.2. On-premises Architecture 8
2.3. Storage Concepts 9

Part III. Configure and Verify **14**

Chapter 3. Considerations for Red Hat Gluster Storage **15**
3.1. Verifying Port Access 15
3.2. Feature Compatibility Support 17

Chapter 4. Adding Servers to the Trusted Storage Pool **19**
4.1. Adding Servers to the Trusted Storage Pool 19
4.2. Removing Servers from the Trusted Storage Pool 21

Chapter 5. Setting Up Storage Volumes **23**
5.1. Setting up Gluster Storage Volumes using gdeploy 24
5.2. Managing Volumes using Heketi 63
5.3. About Encrypted Disk 73
5.4. Formatting and Mounting Bricks 74
5.5. Creating Distributed Volumes 78
5.6. Creating Replicated Volumes 79
5.7. Creating Distributed Replicated Volumes 85
5.8. Creating Arbitrated Replicated Volumes 89
5.9. Creating Dispersed Volumes 99
5.10. Creating Distributed Dispersed Volumes 102
5.11. Starting Volumes 104

Chapter 6. Creating Access to Volumes **105**
6.1. Native Client 105
6.2. NFS 115
6.3. SMB 157
6.4. POSIX Access Control Lists 174
6.5. Managing Object Store 178
6.6. Checking Client Operating Versions 196

Chapter 7. Integrating Red Hat Gluster Storage with Windows Active Directory **198**
7.1. Prerequisites 199
7.2. Integration 200

Part IV. Manage **208**

Chapter 8. Managing Snapshots **209**
8.1. Prerequisites 210
8.2. Creating Snapshots 211
8.3. Cloning a Snapshot 213

8.4. Listing of Available Snapshots 214
8.5. Getting Information of all the Available Snapshots 215
8.6. Getting the Status of Available Snapshots 215
8.7. Configuring Snapshot Behavior 216
8.8. Activating and Deactivating a Snapshot 218
8.9. Deleting Snapshot 218
8.10. Restoring Snapshot 219
8.11. Accessing Snapshots 221
8.12. Scheduling of Snapshots 222
8.13. User Serviceable Snapshots 226
8.14. Troubleshooting 228

Chapter 9. Managing Directory Quotas 233
9.1. Enabling and Disabling Quotas 233
9.2. Before Setting a Quota on a Directory 233
9.3. Limiting Disk Usage 234

Chapter 10. Managing Geo-replication 238
10.1. About Geo-replication 238
10.2. Replicated Volumes vs Geo-replication 238
10.3. Preparing to Deploy Geo-replication 238
10.4. Starting Geo-replication 246
10.5. Starting Geo-replication on a Newly Added Brick, Node, or Volume 255
10.6. Scheduling Geo-replication as a Cron Job 257
10.7. Disaster Recovery 258
10.8. Creating a Snapshot of Geo-replicated Volume 262
10.9. Example - Setting up Cascading Geo-replication 262
10.10. Recommended Practices 264
10.11. Troubleshooting Geo-replication 266

Chapter 11. Managing Red Hat Gluster Storage Volumes 269
11.1. Configuring Volume Options 269
11.2. Setting Multiple Volume Option 269
11.3. Supported Volume Options 270
11.4. Configuring a volume to be mounted read-only 293
11.5. Configuring Transport Types for a Volume 293
11.6. Reserving Storage on a Volume 294
11.7. Expanding Volumes 294
11.8. Shrinking Volumes 298
11.9. Migrating Volumes 304
11.10. Replacing Hosts 312
11.11. Rebalancing Volumes 321
11.12. Setting up Shared Storage Volume 325
11.13. Stopping Volumes 326
11.14. Deleting Volumes 326
11.15. Managing Split-brain 327
11.16. Recommended Configurations - Dispersed Volume 352

Chapter 12. Managing Red Hat Gluster Storage Logs 359
12.1. Log Rotation 359
12.2. Red Hat Gluster Storage Component Logs and Location 359
12.3. Configuring the Log Format 360
12.4. Configuring the Log Level 362
12.5. Suppressing Repetitive Log Messages 364

12.6. Geo-replication Logs 365

Chapter 13. Managing Red Hat Gluster Storage Volume Life-Cycle Extensions 367

13.1. Location of Scripts 367

13.2. Prepackaged Scripts 368

Chapter 14. Managing Containerized Red Hat Gluster Storage 369

14.1. Prerequisites 369

14.2. Starting a Container 369

14.3. Creating a Trusted Storage Pool 371

14.4. Creating a Volume 371

14.5. Mounting a Volume 372

Chapter 15. Detecting BitRot 373

15.1. Enabling and Disabling the BitRot daemon 373

15.2. Modifying BitRot Detection Behavior 373

15.3. Restoring a bad file 374

Chapter 16. Incremental Backup Assistance using Glusterfind 377

16.1. Glusterfind Configuration Options 377

Chapter 17. Managing Tiering 381

17.1. Tiering Architecture 382

17.2. Key Benefits of Tiering 382

17.3. Tiering Limitations 383

17.4. Attaching a Tier to a Volume 383

17.5. Configuring a Tiering Volume 385

17.6. Displaying Tiering Status Information 387

17.7. Detaching a Tier from a Volume 387

Part V. Monitor and Tune 391

Chapter 18. Monitoring Red Hat Gluster Storage 392

18.1. Prerequisites 394

18.2. Installing Nagios 394

18.3. Monitoring Red Hat Gluster Storage Trusted Storage Pool 396

18.4. Monitoring Notifications 415

18.5. Nagios Advanced Configuration 419

18.6. Configuring Nagios Manually 422

18.7. Troubleshooting Nagios 427

Chapter 19. Monitoring Red Hat Gluster Storage Gluster Workload 434

19.1. Profiling volumes 434

19.2. Running the Volume Top Command 437

19.3. gstatus Command 443

19.4. Listing Volumes 447

19.5. Displaying Volume Information 447

19.6. Obtaining Node Information 448

19.7. Retrieving Current Volume Option Settings 464

19.8. Viewing complete volume state with statedump 465

19.9. Displaying Volume Status 468

19.10. Troubleshooting issues in the Red Hat Gluster Storage Trusted Storage Pool 473

Chapter 20. Managing Resource Usage 474

Chapter 21. Tuning for Performance 478

21.1. Disk Configuration 478

21.1. Disk Configuration 478
21.2. Brick Configuration 478
21.3. Network 487
21.4. Memory 487
21.5. Small File Performance Enhancements 487
21.6. Replication 489
21.7. Directory Operations 490
21.8. LVM Cache for Red Hat Gluster Storage 490

Chapter 22. Nagios Configuration Files 496

Part VI. Security 498

Chapter 23. Configuring Network Encryption in Red Hat Gluster Storage 499
23.1. Preparing Certificates 499
23.2. Configuring Network Encryption for a New Trusted Storage Pool 502
23.3. Configuring Network Encryption for an existing Trusted Storage Pool 504
23.4. Enabling Management Encryption 505
23.5. Expanding Volumes 508
23.6. Authorizing a New Client 511
23.7. Deauthorizing a Client 515
23.8. Disabling Network Encryption 516

Part VII. Troubleshoot 519

Chapter 24. Resolving Common Issues 520
24.1. Identifying locked file and clear locks 520
24.2. Retrieving File Path from the Gluster Volume 523
24.3. Resolving glusterd Crash 527
24.4. Restarting a dead/failed brick 527

Part VIII. Appendices 528

Chapter 25. Starting and Stopping the glusterd service 529

Chapter 26. Manually Recovering File Split-brain 530

Appendix A. Revision History 535

Part I. Preface

Chapter 1. Preface

1.1. About Red Hat Gluster Storage

Red Hat Gluster Storage is a software-only, scale-out storage solution that provides flexible and agile unstructured data storage for the enterprise.

Red Hat Gluster Storage provides new opportunities to unify data storage and infrastructure, increase performance, and improve availability and manageability in order to meet a broader set of an organization’s storage challenges and needs.

The product can be installed and managed on-premises, or in a public cloud.

1.2. About glusterFS

glusterFS aggregates various storage servers over network interconnects into one large parallel network file system. Based on a stackable user space design, it delivers exceptional performance for diverse workloads and is a key building block of Red Hat Gluster Storage.

The POSIX compatible glusterFS servers, which use XFS file system format to store data on disks, can be accessed using industry-standard access protocols including Network File System (NFS) and Server Message Block (SMB) (also known as CIFS).

1.3. About On-premises Installation

Red Hat Gluster Storage for On-Premise allows physical storage to be utilized as a virtualized, scalable, and centrally managed pool of storage.

Red Hat Gluster Storage can be installed on commodity servers resulting in a powerful, massively scalable, and highly available NAS environment.

Part II. Overview

Chapter 2. Architecture and Concepts

This chapter provides an overview of Red Hat Gluster Storage architecture and Storage concepts.

2.1. Architecture

At the core of the Red Hat Gluster Storage design is a completely new method of architecting storage. The result is a system that has immense scalability, is highly resilient, and offers extraordinary performance.

In a scale-out system, one of the biggest challenges is keeping track of the logical and physical locations of data and metadata. Most distributed systems solve this problem by creating a metadata server to track the location of data and metadata. As traditional systems add more files, more servers, or more disks, the central metadata server becomes a performance bottleneck, as well as a central point of failure.

Unlike other traditional storage solutions, Red Hat Gluster Storage does not need a metadata server, and locates files algorithmically using an elastic hashing algorithm. This no-metadata server architecture ensures better performance, linear scalability, and reliability.

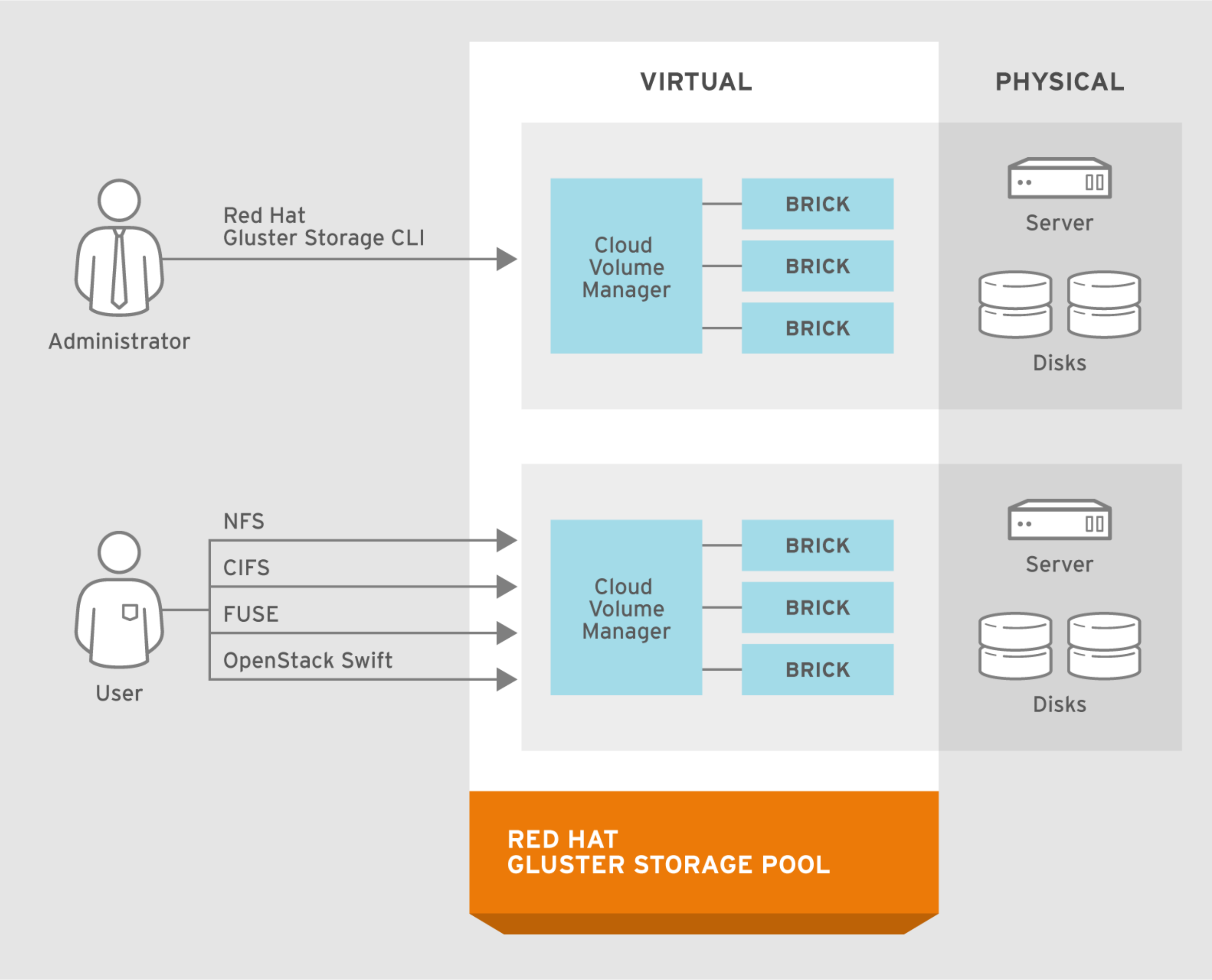


Figure 2.1. Red Hat Gluster Storage Architecture

2.2. On-premises Architecture

Red Hat Gluster Storage for On-premises enables enterprises to treat physical storage as a virtualized, scalable, and centrally managed storage pool by using commodity storage hardware.

It supports multi-tenancy by partitioning users or groups into logical volumes on shared storage. It enables users to eliminate, decrease, or manage their dependence on high-cost, monolithic and difficult-to-deploy storage arrays.

You can add capacity in a matter of minutes across a wide variety of workloads without affecting performance. Storage can also be centrally managed across a variety of workloads, thus increasing storage efficiency.

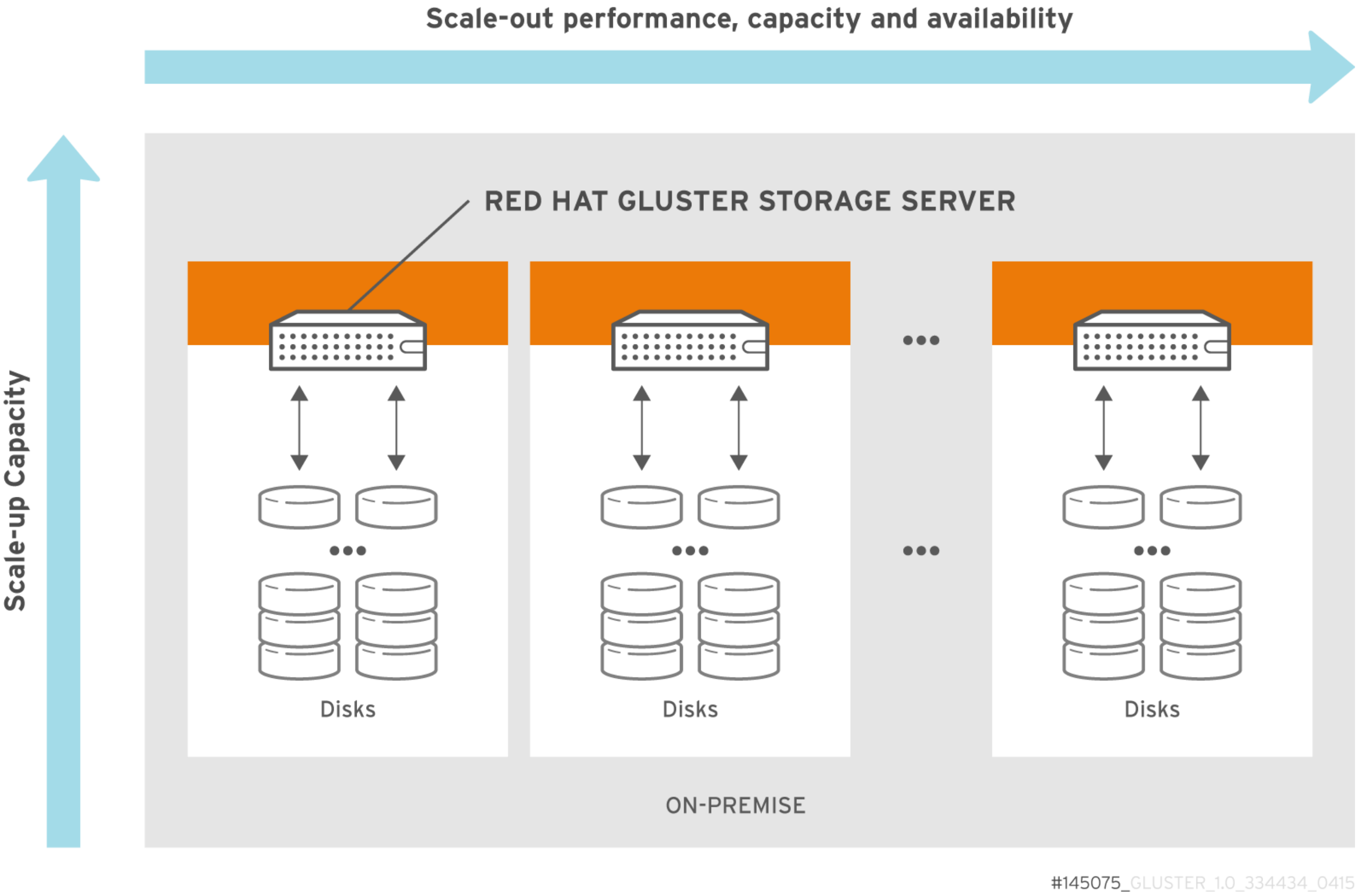

Scale-out performance, capacity and availability
Scale-up Capacity
RED HAT GLUSTER STORAGE SERVER
Disks
Disks
Disks
ON-PREMISE
#145075_GLUSTER_1.0_334434_0415

Figure 2.2. Red Hat Gluster Storage for On-premises Architecture

Red Hat Gluster Storage for On-premises is based on glusterFS, an open source distributed file system with a modular, stackable design, and a unique no-metadata server architecture. This no-metadata server architecture ensures better performance, linear scalability, and reliability.

2.3. Storage Concepts

Following are the common terms relating to file systems and storage used throughout the *Red Hat Gluster Storage Administration Guide*.

Brick

The glusterFS basic unit of storage, represented by an export directory on a server in the trusted storage pool. A brick is expressed by combining a server with an export directory in the following format:

SERVER:EXPORT

For example:

myhostname*:*/exports/myexportdir/

Volume

A volume is a logical collection of bricks. Most of the Red Hat Gluster Storage management operations happen on the volume.

Translator

A translator connects to one or more subvolumes, does something with them, and offers a subvolume connection.

Subvolume

A brick after being processed by at least one translator.

Volfile

Volume (vol) files are configuration files that determine the behavior of your Red Hat Gluster Storage trusted storage pool. At a high level, GlusterFS has three entities, that is, Server, Client and Management daemon. Each of these entities have their own volume files. Volume files for servers and clients are generated by the management daemon upon creation of a volume.

Server and Client Vol files are located in **/var/lib/glusterd/vols/VOLNAME** directory. The management daemon vol file is named as **glusterd.vol** and is located in **/etc/glusterfs/** directory.

Warning

You must not modify any vol file in **/var/lib/glusterd** manually as Red Hat does not support vol files that are not generated by the management daemon.

glusterd

glusterd is the glusterFS Management Service that must run on all servers in the trusted storage pool.

Cluster

A trusted pool of linked computers working together, resembling a single computing resource. In Red Hat Gluster Storage, a cluster is also referred to as a trusted storage pool.

Client

The machine that mounts a volume (this may also be a server).

File System

A method of storing and organizing computer files. A file system organizes files into a database for the storage, manipulation, and retrieval by the computer's operating system.

Source: Wikipedia

Distributed File System

A file system that allows multiple clients to concurrently access data which is spread across servers/bricks in a trusted storage pool. Data sharing among multiple locations is fundamental to all distributed file systems.

Virtual File System (VFS)

VFS is a kernel software layer that handles all system calls related to the standard Linux file system. It provides a common interface to several kinds of file systems.

POSIX

Portable Operating System Interface (for Unix) (POSIX) is the name of a family of related standards specified by the IEEE to define the application programming interface (API), as well as shell and utilities interfaces, for software that is compatible with variants of the UNIX operating system. Red Hat Gluster Storage exports a fully POSIX compatible file system.

Metadata

Metadata is data providing information about other pieces of data.

FUSE

Filesystem in User space (FUSE) is a loadable kernel module for Unix-like operating systems that lets non-privileged users create their own file systems without editing kernel code. This is achieved by running file system code in user space while the FUSE module provides only a "bridge" to the kernel interfaces.

Source: Wikipedia

Geo-Replication

Geo-replication provides a continuous, asynchronous, and incremental replication service from one site to another over Local Area Networks (LAN), Wide Area Networks (WAN), and the Internet.

N-way Replication

Local synchronous data replication that is typically deployed across campus or Amazon Web Services Availability Zones.

Petabyte

A petabyte is a unit of information equal to one quadrillion bytes, or 1000 terabytes. The unit symbol for the petabyte is PB. The prefix peta- (P) indicates a power of 1000:

1 PB = 1,000,000,000,000,000 B = 1000^5 B = 10^15 B.

The term "pebibyte" (PiB), using a binary prefix, is used for the corresponding power of 1024.

Source: Wikipedia

RAID

Redundant Array of Independent Disks (RAID) is a technology that provides increased storage reliability through redundancy. It combines multiple low-cost, less-reliable disk drives components into a logical unit where all drives in the array are interdependent.

RRDNS

Round Robin Domain Name Service (RRDNS) is a method to distribute load across application servers. RRDNS is implemented by creating multiple records with the same name and different IP addresses in the zone file of a DNS server.

Server

The machine (virtual or bare metal) that hosts the file system in which data is stored.

Block Storage

Block special files, or block devices, correspond to devices through which the system moves data in the form of blocks. These device nodes often represent addressable devices such as hard disks, CD-ROM drives, or memory regions. As of Red Hat Gluster Storage 3.4, block storage supports only Container-Native Storage (CNS) and Container-Ready Storage (CRS) use cases. Block storage can be created and configured for this use case by using the **`gluster-block`** command line tool. For more information, see *Container-Native Storage for OpenShift Container Platform*.

Scale-Up Storage

Increases the capacity of the storage device in a single dimension. For example, adding additional disk capacity in a trusted storage pool.

Scale-Out Storage

Increases the capability of a storage device in single dimension. For example, adding more systems of the same size, or adding servers to a trusted storage pool that increases CPU, disk capacity, and throughput for the trusted storage pool.

Trusted Storage Pool

A storage pool is a trusted network of storage servers. When you start the first server, the storage pool consists of only that server.

Namespace

An abstract container or environment that is created to hold a logical grouping of unique identifiers or symbols. Each Red Hat Gluster Storage trusted storage pool exposes a single namespace as a POSIX mount point which contains every file in the trusted storage pool.

User Space

Applications running in user space do not directly interact with hardware, instead using the kernel to moderate access. User space applications are generally more portable than applications in kernel space. glusterFS is a user space application.

Distributed Hash Table Terminology

Hashed subvolume

A Distributed Hash Table Translator subvolume to which the file or directory name is hashed to.

Cached subvolume

A Distributed Hash Table Translator subvolume where the file content is actually present. For directories, the concept of cached-subvolume is not relevant. It is loosely used to mean subvolumes which are not hashed-subvolume.

Linkto-file

For a newly created file, the hashed and cached subvolumes are the same. When directory entry operations like rename (which can change the name and hence hashed subvolume of the file) are performed on the file, instead of moving the entire data in the file to a new hashed subvolume, a file is created with the same name on the newly hashed subvolume. The purpose of this file is only to act as a pointer to the node where the data is present. In the extended attributes of this file, the name of the cached subvolume is stored. This file on the newly hashed-subvolume is called a linkto-file. The linkto file is relevant only for non-directory entities.

Directory Layout

The directory layout helps determine where files in a gluster volume are stored.

When a client creates or requests a file, the DHT translator hashes the file's path to create an integer. Each directory in a gluster subvolume holds files that have integers in a specific range, so the hash of any given file maps to a specific subvolume in the gluster volume. The directory layout determines which integer ranges are assigned to a given directory across all subvolumes.

Directory layouts are assigned when a directory is first created, and can be reassigned by running a rebalance operation on the volume. If a brick or subvolume is offline when a directory is created, it will not be part of the layout until after a rebalance is run.

You should rebalance a volume to recalculate its directory layout after bricks are added to the volume. See Section 11.11, “Rebalancing Volumes” for more information.

Fix Layout

A command that is executed during the rebalance process.

The rebalance process itself comprises of two stages:

1. Fixes the layouts of directories to accommodate any subvolumes that are added or removed. It also heals the directories, checks whether the layout is non-contiguous, and persists the layout in extended attributes, if needed. It also ensures that the directories have the same attributes across all the subvolumes.

2. Migrates the data from the cached-subvolume to the hashed-subvolume.

Part III. Configure and Verify

Chapter 3. Considerations for Red Hat Gluster Storage

3.1. Verifying Port Access

This chapter provides information on the ports that must be open for Red Hat Gluster Storage Server and the **glusterd** service.

The Red Hat Gluster Storage glusterFS daemon **glusterd** enables dynamic configuration changes to Red Hat Gluster Storage volumes, without needing to restart servers or remount storage volumes on clients.

Red Hat Gluster Storage Server uses the listed ports. You must ensure that the firewall settings do not prevent access to these ports.

Firewall configuration tools differ between Red Hat Entperise Linux 6 and Red Hat Enterprise Linux 7.

For Red Hat Enterprise Linux 6, use the **iptables** command to open a port:

```
# iptables -A INPUT -m state --state NEW -m tcp -p tcp --dport 5667 -j
ACCEPT
# service iptables save
```

For Red Hat Enterprise Linux 7, if default ports are in use, it is usually simpler to add a service rather than open a port:

```
# firewall-cmd --zone=zone_name --add-service=glusterfs
# firewall-cmd --zone=zone_name --add-service=glusterfs --permanent
```

However, if the default ports are already in use, you can open a specific port with the following command:

```
# firewall-cmd --zone=zone_name --add-port=port/protocol
# firewall-cmd --zone=zone_name --add-port=port/protocol --permanent
```

For example:

```
# firewall-cmd --zone=public --add-port=5667/tcp
# firewall-cmd --zone=public --add-port=5667/tcp --permanent
```

Table 3.1. TCP Port Numbers

Port Number	Usage
22	For sshd used by geo-replication.
111	For rpc port mapper.
139	For netbios service.
445	For CIFS protocol.
2049	For glusterFS's exports (nfsd process).
24007	For glusterd (for management).
24008	For glusterd RDMA port management.
24009	For glustereventsd.
38465	For gluster-NFS mount protocol.
38466	For gluster-NFS mount protocol.
38468	For gluster-NFS Network Lock Manager (NLM).

Port Number	Usage
38469	For gluster-NFS's ACL support.
39543	For oVirt (Red Hat Gluster Storage Console).
49152 - 49664	Each brick on a host requires its own port for communication with GlusterFS clients. One port is used for each brick, in ascending order, beginning with port 49152. By default, up to 512 ports will be used. The maximum port number can be increased by editing the port range configuration. For more information, see Port Range Configuration. Consider the maximum number of bricks that you expect to exist simultaneously on your host, and open only the ports that you need for your current configuration to help ensure system security.
54321	For VDSM (Red Hat Gluster Storage Console).
55863	For oVirt (Red Hat Gluster Storage Console).

Table 3.2. TCP Port Numbers used for Geo-replication

Port Number	Usage
22	For sshd used by geo-replication.
2049	For glusterFS's exports (nfsd process).
24007	For glusterd (for management).
24008	For glusterd RDMA port management.
49152 - 49664	Each brick on a host requires its own port for communication with GlusterFS clients. One port is used for each brick, in ascending order, beginning with port 49152. By default, up to 512 ports will be used. The maximum port number can be increased by editing the port range configuration. For more information, see Port Range Configuration. Consider the maximum number of bricks that you expect to exist simultaneously on your host, and open only the ports that you need for your current configuration to help ensure system security.

Table 3.3. TCP Port Numbers used for Object Storage (Swift)

Port Number	Usage
443	For HTTPS request.
6010	For Object Server.
6011	For Container Server.
6012	For Account Server.
8080	For Proxy Server.

Table 3.4. TCP Port Numbers for Nagios Monitoring

Port Number	Usage
80	For HTTP protocol (required only if Nagios server is running on a Red Hat Gluster Storage node).
443	For HTTPS protocol (required only for Nagios server).

Port Number	Usage
5667	For NSCA service (required only if Nagios server is running on a Red Hat Gluster Storage node).
5666	For NRPE service (required in all Red Hat Gluster Storage nodes).

Table 3.5. UDP Port Numbers

Port Number	Usage
111	For RPC Bind.

For more information regarding port and firewall details for NFS-Ganesha, refer Section 6.2.3.2.1, “Port and Firewall Information for NFS-Ganesha”

For more information regarding port and firewall details for Red Hat Gluster Storage Web Administration, refer Section 2.4. Firewall Configuration in *Quick Start Guide*.

3.2. Feature Compatibility Support

Red Hat Gluster Storage supports a number of features. Most features are supported with other features, but there are some exceptions. This section clearly identifies which features are supported and compatible with other features to help you in planning your Red Hat Gluster Storage deployment.

Note

Internet Protocol Version 6 (IPv6) support is available only for Red Hat Hyperconverged Infrastructure for Virtualization environments and not for Red Hat Gluster Storage standalone environments.

Features in the following table are supported from the specified version and later.

Table 3.6. Features supported by Red Hat Gluster Storage version

Feature	Version
Arbiter bricks	3.2
Bitrot detection	3.1
Erasure coding	3.1
Google Compute Engine	3.1.3
Metadata caching	3.2
Microsoft Azure	3.1.3
NFS version 4	3.1
SELinux	3.1
Sharding	3.2.0
Snapshots	3.0
Snapshots, cloning	3.1.3
Snapshots, user-serviceable	3.0.3
Tiering	3.1.2
Volume Shadow Copy (VSS)	3.1.3

Table 3.7. Features supported by volume type

Volume Type	Sharding	Tiering	Quota	Snapshots	Geo-Rep	Bitrot
Arbitrated-Replicated	Yes	No	Yes	Yes	Yes	Yes
Distributed	No	Yes	Yes	Yes	Yes	Yes
Distributed-Dispersed	No	Yes	Yes	Yes	Yes	Yes
Distributed-Replicated	Yes	Yes	Yes	Yes	Yes	Yes
Replicated	Yes	Yes	Yes	Yes	Yes	Yes
Sharded	N/A	No	No	No	Yes	No
Tiered	No	N/A	Limited [a]	Limited[a]	Limited[a]	Limited[a]

[a] See Section 17.3. Tiering Limitations in the *Red Hat Gluster Storage 3.4 Administration Guide* for details.

Table 3.8. Features supported by client protocol

Feature	FUSE	Gluster-NFS	NFS-Ganesha	SMB	Swift/S3
Arbiter	Yes	Yes	Yes	Yes	No
Bitrot detection	Yes	Yes	No	Yes	No
dm-cache	Yes	Yes	Yes	Yes	Yes
Encryption (TLS-SSL)	Yes	Yes	Yes	Yes	No
Erasure coding	Yes	Yes	Yes	Yes	No
Export subdirectory	Yes	Yes	Yes	N/A	N/A
Geo-replication	Yes	Yes	Yes	Yes	Yes
Quota	Yes	Yes	Yes	Yes	No
RDMA	Yes	No	No	No	N/A
Snapshots	Yes	Yes	Yes	Yes	Yes
Snapshot cloning	Yes	Yes	Yes	Yes	Yes
Tiering	Yes	Yes	N/A	N/A	N/A

Chapter 4. Adding Servers to the Trusted Storage Pool

A storage pool is a network of storage servers.

When the first server starts, the storage pool consists of that server alone. Adding additional storage servers to the storage pool is achieved using the probe command from a running, trusted storage server.

Important

Before adding servers to the trusted storage pool, you must ensure that the ports specified in Chapter 3, *Considerations for Red Hat Gluster Storage* are open.

On Red Hat Enterprise Linux 7, enable the glusterFS firewall service in the active zones for runtime and permanent mode using the following commands:

To get a list of active zones, run the following command:

```
# firewall-cmd --get-active-zones
```

To allow the firewall service in the active zones, run the following commands:

```
# firewall-cmd --zone=zone_name --add-service=glusterfs
# firewall-cmd --zone=zone_name --add-service=glusterfs --permanent
```

For more information about using firewalls, see section *Using Firewalls* in the *Red Hat Enterprise Linux 7 Security Guide*: https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/Security_Guide/sec-Using_Firewalls.html.

Note

When any two gluster commands are executed concurrently on the same volume, the following error is displayed:

Another transaction is in progress.

This behavior in the Red Hat Gluster Storage prevents two or more commands from simultaneously modifying a volume configuration, potentially resulting in an inconsistent state. Such an implementation is common in environments with monitoring frameworks such as the Red Hat Gluster Storage Console, Red Hat Enterprise Virtualization Manager, and Nagios. For example, in a four node Red Hat Gluster Storage Trusted Storage Pool, this message is observed when **`gluster volume status`** ***`VOLNAME`*** command is executed from two of the nodes simultaneously.

4.1. Adding Servers to the Trusted Storage Pool

The **`gluster peer probe`** ***`[server]`*** command is used to add servers to the trusted server pool.

Note

Probing a node from lower version to a higher version of Red Hat Gluster Storage node is not supported.

Adding Three Servers to a Trusted Storage Pool

Create a trusted storage pool consisting of three storage servers, which comprise a volume.

Prerequisites

- The **`glusterd`** service must be running on all storage servers requiring addition to the trusted storage pool. See Chapter 25, *Starting and Stopping the glusterd service* for service start and stop commands.
- **`Server1`**, the trusted storage server, is started.
- The host names of the target servers must be resolvable by DNS.

1. Run **`gluster peer probe`** ***`[server]`*** from Server 1 to add additional servers to the trusted storage pool.

   Note

   - Self-probing **`Server1`** will result in an error because it is part of the trusted storage pool by default.
   - All the servers in the Trusted Storage Pool must have RDMA devices if either **`RDMA`** or **`RDMA,TCP`** volumes are created in the storage pool. The peer probe must be performed using IP/hostname assigned to the RDMA device.

   ```
   # gluster peer probe server2
   Probe successful

   # gluster peer probe server3
   Probe successful

   # gluster peer probe server4
   Probe successful
   ```

2. Verify the peer status from all servers using the following command:

   ```
   # gluster peer status
     Number of Peers: 3

     Hostname: server2
     Uuid: 5e987bda-16dd-43c2-835b-08b7d55e94e5
     State: Peer in Cluster (Connected)

     Hostname: server3
     Uuid: 1e0ca3aa-9ef7-4f66-8f15-cbc348f29ff7
     State: Peer in Cluster (Connected)
   ```

```
Hostname: server4
Uuid: 3e0caba-9df7-4f66-8e5d-cbc348f29ff7
State: Peer in Cluster (Connected)
```

Important

If the existing trusted storage pool has a geo-replication session, then after adding the new server to the trusted storage pool, perform the steps listed at Section 10.5, “Starting Geo-replication on a Newly Added Brick, Node, or Volume”.

Note

Verify that time is synchronized on all Gluster nodes by using the following command:

```
# for peer in `gluster peer status | grep Hostname | awk -F':' '{print
$2}' | awk '{print $1}'`; do clockdiff $peer; done
```

4.2. Removing Servers from the Trusted Storage Pool

Warning

Before detaching a peer from the trusted storage pool, make sure that the clients are not using the node. If backup servers were not set at mount time using the backup-volfile-servers option, remount the volume on the client using the IP address or FQDN of another server in the trusted storage pool to avoid inconsistencies.

Run **`gluster peer detach`** ***`server`*** to remove a server from the storage pool.

Removing One Server from the Trusted Storage Pool

Remove one server from the Trusted Storage Pool, and check the peer status of the storage pool.

Prerequisites

- The **`glusterd`** service must be running on the server targeted for removal from the storage pool. See Chapter 25, *Starting and Stopping the glusterd service* for service start and stop commands.
- The host names of the target servers must be resolvable by DNS.

1. Run **`gluster peer detach`** ***`[server]`*** to remove the server from the trusted storage pool.

```
# gluster peer detach server4
Detach successful
```

2. Verify the peer status from all servers using the following command:

```
# gluster peer status
Number of Peers: 2

Hostname: server2
Uuid: 5e987bda-16dd-43c2-835b-08b7d55e94e5
State: Peer in Cluster (Connected)

Hostname: server3
Uuid: 1e0ca3aa-9ef7-4f66-8f15-cbc348f29ff7
```

Chapter 5. Setting Up Storage Volumes

A Red Hat Gluster Storage volume is a logical collection of bricks, where each brick is an export directory on a server in the trusted storage pool. Most of the Red Hat Gluster Storage Server management operations are performed on the volume. For a detailed information about configuring Red Hat Gluster Storage for enhancing performance see, Chapter 21, *Tuning for Performance*

Warning

Red Hat does not support writing data directly into the bricks. Read and write data only through the Native Client, or through NFS or SMB mounts.

Note

Red Hat Gluster Storage supports IP over Infiniband (IPoIB). Install Infiniband packages on all Red Hat Gluster Storage servers and clients to support this feature. Run the `yum groupinstall "Infiniband Support"` to install Infiniband packages.

Volume Types

Distributed

Distributes files across bricks in the volume.

Use this volume type where scaling and redundancy requirements are not important, or provided by other hardware or software layers.

See Section 5.5, “Creating Distributed Volumes” for additional information about this volume type.

Replicated

Replicates files across bricks in the volume.

Use this volume type in environments where high-availability and high-reliability are critical.

See Section 5.6, “Creating Replicated Volumes” for additional information about this volume type.

Distributed Replicated

Distributes files across replicated bricks in the volume.

Use this volume type in environments where high-reliability and scalability are critical. This volume type offers improved read performance in most environments.

See Section 5.7, “Creating Distributed Replicated Volumes” for additional information about this volume type.

Arbitrated Replicated

Replicates files across two bricks in a replica set, and replicates only metadata to the third brick.

Use this volume type in environments where consistency is critical, but underlying storage space is at a premium.

See Section 5.8, “Creating Arbitrated Replicated Volumes” for additional information about this volume type.

Dispersed

Disperses the file's data across the bricks in the volume.

Use this volume type where you need a configurable level of reliability with a minimum space waste.

See Section 5.9, “Creating Dispersed Volumes” for additional information about this volume type.

Distributed Dispersed

Distributes file's data across the dispersed sub-volume.

Use this volume type where you need a configurable level of reliability with a minimum space waste.

See Section 5.10, “Creating Distributed Dispersed Volumes” for additional information about this volume type.

5.1. Setting up Gluster Storage Volumes using gdeploy

The gdeploy tool automates the process of creating, formatting, and mounting bricks. With gdeploy, the manual steps listed between *Section 5.4 Formatting and Mounting Bricks* and *Section 5.10 Creating Distributed Dispersed Volumes* are automated.

When setting-up a new trusted storage pool, gdeploy could be the preferred choice of trusted storage pool set up, as manually executing numerous commands can be error prone.

The advantages of using gdeploy to automate brick creation are as follows:

- Setting-up the backend on several machines can be done from one's laptop/desktop. This saves time and scales up well when the number of nodes in the trusted storage pool increase.
- Flexibility in choosing the drives to configure. (sd, vd, ...).
- Flexibility in naming the logical volumes (LV) and volume groups (VG).

5.1.1. Getting Started

Prerequisites

1. Generate the passphrase-less SSH keys for the nodes which are going to be part of the trusted storage pool by running the following command:

```
# ssh-keygen -t rsa -N ''
```

2. Set up key-based SSH authentication access between the gdeploy controller and servers by running the following command:

```
# ssh-copy-id -i root@server
```

> **Note**
>
> If you are using a Red Hat Gluster Storage node as the deployment node and not an external node, then the key-based SSH authentication must be set up for the Red Hat Gluster Storage node from where the installation is performed.

3. Enable the repository required to install Ansible by running the following command:

   ```
   # subscription-manager repos --enable=rhel-7-server-ansible-2-rpms
   ```

4. Install **`ansible`** by executing the following command:

   ```
   # yum install ansible
   ```

5. You must also ensure the following:

   - Devices should be raw and unused
   - Default system locale must be set to **`en_US`**

     For information on system locale, refer to the *Setting the System Locale* of the *Red Hat Enterprise Linux 7 System Administrator's Guide*.

   - For multiple devices, use multiple volume groups, thinpool, and thinvol in the **`gdeploy`** configuration file

For more information, see Installing Ansible to Support Gdeploy in *Red Hat Gluster Storage 3.4 Installation Guide*.

gdeploy can be used to deploy Red Hat Gluster Storage in two ways:

- Using a node in a trusted storage pool
- Using a machine outside the trusted storage pool

Using a node in a cluster

The **`gdeploy`** package is bundled as part of the initial installation of Red Hat Gluster Storage.

Using a machine outside the trusted storage pool

You must ensure that the Red Hat Gluster Storage is subscribed to the required channels. For more information see, *Subscribing to the Red Hat Gluster Storage Server Channels*in the *Red Hat Gluster Storage 3.4 Installation Guide*.

Execute the following command to install gdeploy:

```
# yum install gdeploy
```

For more information on installing **`gdeploy`** see, *Installing Ansible to Support Gdeploy* section in the *Red Hat Gluster Storage 3.4 Installation Guide*.

5.1.2. Setting up a Trusted Storage Pool

Creating a trusted storage pool is a tedious task and becomes more tedious as the nodes in the trusted storage pool grow. With gdeploy, just a configuration file can be used to set up a trusted storage pool. When gdeploy is installed, a sample configuration file will be created at:

```
/usr/share/doc/gdeploy/examples/gluster.conf.sample
```

Note

The trusted storage pool can be created either by performing each tasks, such as, setting up a backend, creating a volume, and mounting volumes independently or summed up as a single configuration.

For example, for a basic trusted storage pool of a 2 x 2 replicated volume the configuration details in the configuration file will be as follows:

2x2-volume-create.conf:

```
#
# Usage:
#       gdeploy -c 2x2-volume-create.conf
#
# This does backend setup first and then create the volume using the
# setup bricks.
#
#

[hosts]
10.70.46.13
10.70.46.17


# Common backend setup for 2 of the hosts.
[backend-setup]
devices=sdb,sdc
vgs=vg1,vg2
pools=pool1,pool2
lvs=lv1,lv2
mountpoints=/rhgs/brick1,/rhgs/brick2
brick_dirs=/rhgs/brick1/b1,/rhgs/brick2/b2

# If backend-setup is different for each host
# [backend-setup:10.70.46.13]
# devices=sdb
# brick_dirs=/rhgs/brick1
#
# [backend-setup:10.70.46.17]
# devices=sda,sdb,sdc
# brick_dirs=/rhgs/brick{1,2,3}
#

[volume]
action=create
volname=sample_volname
```

```
replica=yes
replica_count=2
force=yes


[clients]
action=mount
volname=sample_volname
hosts=10.70.46.15
fstype=glusterfs
client_mount_points=/mnt/gluster
```

With this configuration a 2 x 2 replica trusted storage pool with the given IP addresses and backend device as /dev/sdb,/dev/sdc with the volume name as sample_volname will be created.

For more information on possible values, see Section 5.1.7, “Configuration File”

After modifying the configuration file, invoke the configuration using the command:

```
# gdeploy -c conf.txt
```

Note

You can create a new configuration file by referencing the template file available at **/usr/share/doc/gdeploy/examples/gluster.conf.sample**. To invoke the new configuration file, run **gdeploy -c *path_to_file/config.txt*** command.

To **only** setup the backend see, Section 5.1.3, “Setting up the Backend ”

To **only** create a volume see, Section 5.1.4, “Creating Volumes”

To **only** mount clients see, Section 5.1.5, “Mounting Clients”

5.1.3. Setting up the Backend

In order to setup a Gluster Storage volume, the LVM thin-p must be set up on the storage disks. If the number of machines in the trusted storage pool is huge, these tasks takes a long time, as the number of commands involved are huge and error prone if not cautious. With gdeploy, just a configuration file can be used to set up a backend. The backend is setup at the time of setting up a fresh trusted storage pool, which requires bricks to be setup before creating a volume. When gdeploy is installed, a sample configuration file will be created at:

```
/usr/share/doc/gdeploy/examples/gluster.conf.sample
```

A backend can be setup in two ways:

- Using the [backend-setup] module
- Creating Physical Volume (PV), Volume Group (VG), and Logical Volume (LV) individually

Note

For Red Hat Enterprise Linux 6, the **`xfsprogs`** package must be installed before setting up the backend bricks using gdeploy.

5.1.3.1. Using the [backend-setup] Module

Backend setup can be done on specific machines or on all the machines. The backend-setup module internally creates PV, VG, and LV and mounts the device. Thin-p logical volumes are created as per the performance recommendations by Red Hat.

The backend can be setup based on the requirement, such as:

- Generic
- Specific

Generic

If the disk names are uniform across the machines then backend setup can be written as below. The backend is setup for all the hosts in the `hosts' section.

For more information on possible values, see Section 5.1.7, “Configuration File”

Example configuration file: Backend-setup-generic.conf

```
#
# Usage:
#       gdeploy -c backend-setup-generic.conf
#
# This configuration creates backend for GlusterFS clusters
#

[hosts]
10.70.46.130
10.70.46.32
10.70.46.110
10.70.46.77

# Backend setup for all the nodes in the `hosts' section. This will create
# PV, VG, and LV with gdeploy generated names.
[backend-setup]
devices=vdb
```

Specific

If the disks names vary across the machines in the cluster then backend setup can be written for specific machines with specific disk names. gdeploy is quite flexible in allowing to do host specific setup in a single configuration file.

For more information on possible values, see Section 5.1.7, “Configuration File”

Example configuration file: backend-setup-hostwise.conf

```
#
```

```
# Usage:
#       gdeploy -c backend-setup-hostwise.conf
#
# This configuration creates backend for GlusterFS clusters
#

[hosts]
10.70.46.130
10.70.46.32
10.70.46.110
10.70.46.77

# Backend setup for 10.70.46.77 with default gdeploy generated names for
# Volume Groups and Logical Volumes. Volume names will be GLUSTER_vg1,
# GLUSTER_vg2...
[backend-setup:10.70.46.77]
devices=vda,vdb

# Backend setup for remaining 3 hosts in the `hosts' section with custom
names
# for Volumes Groups and Logical Volumes.
[backend-setup:10.70.46.{130,32,110}]
devices=vdb,vdc,vdd
vgs=vg1,vg2,vg3
pools=pool1,pool2,pool3
lvs=lv1,lv2,lv3
mountpoints=/rhgs/brick1,/rhgs/brick2,/rhgs/brick3
brick_dirs=/rhgs/brick1/b1,/rhgs/brick2/b2,/rhgs/brick3/b3
```

5.1.3.2. Creating Backend by Setting up PV, VG, and LV

If the user needs more control over setting up the backend, then pv, vg, and lv can be created individually. LV module provides flexibility to create more than one LV on a VG. For example, the `backend-setup' module setups up a thin-pool by default and applies default performance recommendations. However, if the user has a different use case which demands more than one LV, and a combination of thin and thick pools then `backend-setup' is of no help. The user can use PV, VG, and LV modules to achieve this.

For more information on possible values, see Section 5.1.7, “Configuration File”

The below example shows how to create four logical volumes on a single volume group. The examples shows a mix of thin and thickpool LV creation.

```
[hosts]
10.70.46.130
10.70.46.32

[pv]
action=create
devices=vdb

[vg1]
action=create
vgname=RHS_vg1
pvname=vdb

[lv1]
```

```
action=create
vgname=RHS_vg1
lvname=engine_lv
lvtype=thick
size=10GB
mount=/rhgs/brick1

[lv2]
action=create
vgname=RHS_vg1
poolname=lvthinpool
lvtype=thinpool
poolmetadatasize=200MB
chunksize=1024k
size=30GB

[lv3]
action=create
lvname=lv_vmaddldisks
poolname=lvthinpool
vgname=RHS_vg1
lvtype=thinlv
mount=/rhgs/brick2
virtualsize=9GB

[lv4]
action=create
lvname=lv_vmrootdisks
poolname=lvthinpool
vgname=RHS_vg1
size=19GB
lvtype=thinlv
mount=/rhgs/brick3
virtualsize=19GB
```

Example to extend an existing VG:

```
#
# Extends a given given VG. pvname and vgname is mandatory, in this example
the
# vg `RHS_vg1' is extended by adding pv, vdd. If the pv is not alreay
present, it
# is created by gdeploy.
#
[hosts]
10.70.46.130
10.70.46.32

[vg2]
action=extend
vgname=RHS_vg1
pvname=vdd
```

5.1.4. Creating Volumes

Setting up volume involves writing long commands by choosing the hostname/IP and brick order carefully and this could be error prone. gdeploy helps in simplifying this task. When gdeploy is installed, a sample configuration file will be created at:

```
/usr/share/doc/gdeploy/examples/gluster.conf.sample
```

For example, for a basic trusted storage pool of a 2 x 2 replicate volume the configuration details in the configuration file will be as follows:

```
[hosts]
10.0.0.1
10.0.0.2
10.0.0.3
10.0.0.4

[volume]
action=create
volname=glustervol
transport=tcp,rdma
replica=yes
replica_count=2
force=yes
```

For more information on possible values, see Section 5.1.7, “Configuration File”

After modifying the configuration file, invoke the configuration using the command:

```
# gdeploy -c conf.txt
```

Creating Multiple Volumes

Note

Support of creating multiple volumes only from gdeploy 2.0, please check your gdeploy version before trying this configuration.

While creating multiple volumes in a single configuration, the [volume] modules should be numbered. For example, if there are two volumes they will be numbered [volume1], [volume2]

vol-create.conf

```
[hosts]
10.70.46.130
10.70.46.32

[backend-setup]
devices=vdb,vdc
mountpoints=/mnt/data1,/mnt/data2

[volume1]
action=create
volname=vol-one
transport=tcp
```

```
replica=yes
replica_count=2
brick_dirs=/mnt/data1/1

[volume2]
action=create
volname=vol-two
transport=tcp
replica=yes
replica_count=2
brick_dirs=/mnt/data2/2
```

With gdeploy 2.0, a volume can be created with multiple volume options set. Number of keys should match number of values.

```
[hosts]
10.70.46.130
10.70.46.32

[backend-setup]
devices=vdb,vdc
mountpoints=/mnt/data1,/mnt/data2

[volume1]
action=create
volname=vol-one
transport=tcp
replica=yes
replica_count=2
key=group,storage.owner-uid,storage.owner-gid,features.shard,features.shard-
block-size,performance.low-prio-threads,cluster.data-self-heal-algorithm
value=virt,36,36,on,512MB,32,full
brick_dirs=/mnt/data1/1

[volume2]
action=create
volname=vol-two
transport=tcp
replica=yes
key=group,storage.owner-uid,storage.owner-gid,features.shard,features.shard-
block-size,performance.low-prio-threads,cluster.data-self-heal-algorithm
value=virt,36,36,on,512MB,32,full
replica_count=2
brick_dirs=/mnt/data2/2
```

The above configuration will create two volumes with multiple volume options set.

5.1.5. Mounting Clients

When mounting clients, instead of logging into every client which has to be mounted, gdeploy can be used to mount clients remotely. When gdeploy is installed, a sample configuration file will be created at:

```
/usr/share/doc/gdeploy/examples/gluster.conf.sample
```

Following is an example of the modifications to the configuration file in order to mount clients:

```
[clients]
action=mount
hosts=10.70.46.159
fstype=glusterfs
client_mount_points=/mnt/gluster
volname=10.0.0.1:glustervol
```

Note

If the **`fstype`** is NFS, then mention it as nfs-version. By default it is 3.

For more information on possible values, see Section 5.1.7, “Configuration File”

After modifying the configuration file, invoke the configuration using the command:

```
# gdeploy -c conf.txt
```

5.1.6. Configuring a Volume

The volumes can be configured using the configuration file. The volumes can be configured remotely using the configuration file without having to log into the trusted storage pool. For more information regarding the sections and options in the configuration file, see Section 5.1.7, “Configuration File”

5.1.6.1. Adding and Removing a Brick

The configuration file can be modified to add or remove a brick:

Adding a Brick

Modify the [volume] section in the configuration file to add a brick. For example:

```
[volume]
action=add-brick
volname=10.0.0.1:glustervol
bricks=10.0.0.1:/rhgs/new_brick
```

After modifying the configuration file, invoke the configuration using the command:

```
# gdeploy -c conf.txt
```

Removing a Brick

Modify the [volume] section in the configuration file to remove a brick. For example:

```
[volume]
action=remove-brick
volname=10.0.0.1:glustervol
bricks=10.0.0.2:/rhgs/brick
state=commit
```

Other options for **`state`** are stop, start, and force.

After modifying the configuration file, invoke the configuration using the command:

```
# gdeploy -c conf.txt
```

For more information on possible values, see Section 5.1.7, “Configuration File”

5.1.6.2. Rebalancing a Volume

Modify the [volume] section in the configuration file to rebalance a volume. For example:

```
[volume]
action=rebalance
volname=10.70.46.13:glustervol
state=start
```

Other options for **`state`** are stop, and fix-layout.

After modifying the configuration file, invoke the configuration using the command:

```
# gdeploy -c conf.txt
```

For more information on possible values, see Section 5.1.7, “Configuration File”

5.1.6.3. Starting, Stopping, or Deleting a Volume

The configuration file can be modified to start, stop, or delete a volume:

Starting a Volume

Modify the [volume] section in the configuration file to start a volume. For example:

```
[volume]
action=start
volname=10.0.0.1:glustervol
```

After modifying the configuration file, invoke the configuration using the command:

```
# gdeploy -c conf.txt
```

Stopping a Volume

Modify the [volume] section in the configuration file to start a volume. For example:

```
[volume]
action=stop
volname=10.0.0.1:glustervol
```

After modifying the configuration file, invoke the configuration using the command:

```
# gdeploy -c conf.txt
```

Deleting a Volume

Modify the [volume] section in the configuration file to start a volume. For example:

```
[volume]
action=delete
volname=10.70.46.13:glustervol
```

After modifying the configuration file, invoke the configuration using the command:

```
# gdeploy -c conf.txt
```

For more information on possible values, see Section 5.1.7, “Configuration File”

5.1.7. Configuration File

The configuration file includes the various options that can be used to change the settings for gdeploy. The following options are currently supported:

- [hosts]
- [devices]
- [disktype]
- [diskcount]
- [stripesize]
- [vgs]
- [pools]
- [lvs]
- [mountpoints]
- [peer]
- [clients]
- [volume]
- [backend-setup]
- [pv]
- [vg]
- [lv]
- [RH-subscription]
- [yum]
- [shell]
- [update-file]
- [service]
- [script]

- [firewalld]

The options are briefly explained in the following list:

- **hosts**

  This is a mandatory section which contains the IP address or hostname of the machines in the trusted storage pool. Each hostname or IP address should be listed in a separate line.

  For example:

  ```
  [hosts]
  10.0.0.1
  10.0.0.2
  ```

- **devices**

  This is a generic section and is applicable to all the hosts listed in the [hosts] section. However, if sections of hosts such as the [hostname] or [IP-address] is present, then the data in the generic sections like [devices] is ignored. Host specific data take precedence. This is an optional section.

  For example:

  ```
  [devices]
  /dev/sda
  /dev/sdb
  ```

  > **Note**
  >
  > When configuring the backend setup, the devices should be either listed in this section or in the host specific section.

- **disktype**

  This section specifies the disk configuration that is used while setting up the backend. gdeploy supports RAID 10, RAID 6, RAID 5, and JBOD configurations. This is an optional section and if the field is left empty, JBOD is taken as the default configuration. Valid values for this field are **`raid10`**, **`raid6`**, **`raid5`**, and **`jbod`**.

  For example:

  ```
  [disktype]
  raid6
  ```

- **diskcount**

  This section specifies the number of data disks in the setup. This is a mandatory field if a RAID disk type is specified under **`[disktype]`**. If the [disktype] is JBOD the [diskcount] value is ignored. This parameter is host specific.

  For example:

```
[diskcount]
10
```

- **stripesize**

  This section specifies the stripe_unit size in KB.

  Case 1: This field is not necessary if the [disktype] is JBOD, and any given value will be ignored.

  Case 2: This is a mandatory field if [disktype] is specified as RAID 5 or RAID 6.

  For [disktype] RAID 10, the default value is taken as 256KB. Red Hat does not recommend changing this value. If you specify any other value the following warning is displayed:

  ```
  "Warning: We recommend a stripe unit size of 256KB for RAID 10"
  ```

  Note

  Do not add any suffixes like K, KB, M, etc. This parameter is host specific and can be added in the hosts section.

  For example:

  ```
  [stripesize]
  128
  ```

- **vgs**

  This section is deprecated in gdeploy 2.0. Please see [backend-setup] for more details for gdeploy 2.0. This section specifies the volume group names for the devices listed in [devices]. The number of volume groups in the [vgs] section should match the one in [devices]. If the volume group names are missing, the volume groups will be named as GLUSTER_vg{1, 2, 3, ...} as default.

  For example:

  ```
  [vgs]
  CUSTOM_vg1
  CUSTOM_vg2
  ```

- **pools**

  This section is deprecated in gdeploy 2.0. Please see [backend-setup] for more details for gdeploy 2.0. This section specifies the pool names for the volume groups specified in the [vgs] section. The number of pools listed in the [pools] section should match the number of volume groups in the [vgs] section. If the pool names are missing, the pools will be named as GLUSTER_pool{1, 2, 3, ...}.

  For example:

  ```
  [pools]
  CUSTOM_pool1
  CUSTOM_pool2
  ```

- **lvs**

  This section is deprecated in gdeploy 2.0. Please see [backend-setup] for more details for gdeploy 2.0. This section provides the logical volume names for the volume groups specified in [vgs]. The number of logical volumes listed in the [lvs] section should match the number of volume groups listed in [vgs]. If the logical volume names are missing, it is named as GLUSTER_lv{1, 2, 3, ...}.

  For example:

  ```
  [lvs]
  CUSTOM_lv1
  CUSTOM_lv2
  ```

- **mountpoints**

  This section is deprecated in gdeploy 2.0. Please see [backend-setup] for more details for gdeploy 2.0. This section specifies the brick mount points for the logical volumes. The number of mount points should match the number of logical volumes specified in [lvs] If the mount points are missing, the mount points will be names as /gluster/brick{1, 2, 3…}.

  For example:

  ```
  [mountpoints]
  /rhgs/brick1
  /rhgs/brick2
  ```

- **peer**

  This section specifies the configurations for the Trusted Storage Pool management (TSP). This section helps in making all the hosts specified in the [hosts] section to either probe each other to create the trusted storage pool or detach all of them from the trusted storage pool. The only option in this section is the option names 'action' which can have it's values to be either probe or detach.

  For example:

  ```
  [peer]
  action=probe
  ```

- **clients**

  This section specifies the client hosts and client_mount_points to mount the gluster storage volume created. The 'action' option is to be specified for the framework to determine the action that has to be performed. The options are 'mount' and 'unmount'. The Client hosts field is mandatory. If the mount points are not specified, default will be taken as /mnt/gluster for all the hosts.

  The option fstype specifies how the gluster volume is to be mounted. Default is glusterfs (FUSE mount). The volume can also be mounted as NFS. Each client can have different types of volume mount, which has to be specified with a comma separated. The following fields are included:

  ```
  * action
  * hosts
  * fstype
  * client_mount_points
  ```

  For example:

```
[clients]
action=mount
hosts=10.0.0.10
fstype=nfs
options=vers=3
client_mount_points=/mnt/rhs
```

- **volume**

  The section specifies the configuration options for the volume. The following fields are included in this section:

  ```
  * action
  * volname
  * transport
  * replica
  * replica_count
  * disperse
  * disperse_count
  * redundancy_count
  * force
  ```

  - **action**

    This option specifies what action must be performed in the volume. The choices can be [create, delete, add-brick, remove-brick].

    create: This choice is used to create a volume.

    delete: If the delete choice is used, all the options other than 'volname' will be ignored.

    add-brick or *remove-brick*: If the add-brick or remove-brick is chosen, extra option bricks with a comma separated list of brick names(in the format <hostname>:<brick path> should be provided. In case of remove-brick, state option should also be provided specifying the state of the volume after brick removal.

  - **volname**

    This option specifies the volume name. Default name is glustervol

    > **Note**
    >
    > - In case of a volume operation, the 'hosts' section can be omitted, provided volname is in the format <hostname>:<volname>, where hostname is the hostname / IP of one of the nodes in the cluster
    > - Only single volume creation/deletion/configuration is supported.

  - **transport**

    This option specifies the transport type. Default is tcp. Options are tcp or rdma or tcp,rdma.

  - **replica**

    This option will specify if the volume should be of type replica. options are yes and no. Default is no. If 'replica' is provided as yes, the 'replica_count' should be provided.

  - **disperse**

    This option specifies if the volume should be of type disperse. Options are yes and no. Default is no.

  - **disperse_count**

    This field is optional even if 'disperse' is yes. If not specified, the number of bricks specified in the command line is taken as the disperse_count value.

  - **redundancy_count**

    If this value is not specified, and if 'disperse' is yes, it's default value is computed so that it generates an optimal configuration.

  - **force**

    This is an optional field and can be used during volume creation to forcefully create the volume.

  For example:

  ```
  [volname]
  action=create
  volname=glustervol
  transport=tcp,rdma
  replica=yes
  replica_count=3
  force=yes
  ```

- **backend-setup**

  Available in gdeploy 2.0. This section sets up the backend for using with GlusterFS volume. If more than one backend-setup has to be done, they can be done by numbering the section like [backend-setup1], [backend-setup2], ...

  backend-setup section supports the following variables:

  - devices: This replaces the [pvs] section in gdeploy 1.x. devices variable lists the raw disks which should be used for backend setup. For example:

    ```
    [backend-setup]
    devices=sda,sdb,sdc
    ```

    This is a mandatory field.

  - dalign:

    The Logical Volume Manager can use a portion of the physical volume for storing its metadata while the rest is used as the data portion. Align the I/O at the Logical Volume Manager (LVM) layer using the dalign option while creating the physical volume. For example:

    ```
    [backend-setup]
    devices=sdb,sdc,sdd,sde
    dalign=256k
    ```

For JBOD, use an alignment value of 256K. For hardware RAID, the alignment value should be obtained by multiplying the RAID stripe unit size with the number of data disks. If 12 disks are used in a RAID 6 configuration, the number of data disks is 10; on the other hand, if 12 disks are used in a RAID 10 configuration, the number of data disks is 6.

The following example is appropriate for 12 disks in a RAID 6 configuration with a stripe unit size of 128 KiB:

```
[backend-setup]
devices=sdb,sdc,sdd,sde
dalign=1280k
```

The following example is appropriate for 12 disks in a RAID 10 configuration with a stripe unit size of 256 KiB:

```
[backend-setup]
devices=sdb,sdc,sdd,sde
dalign=1536k
```

To view the previously configured physical volume settings for the dalign option, run the **pvs -o +pe_start *device*** command. For example:

```
# pvs -o +pe_start disk
PV         VG   Fmt  Attr PSize PFree 1st PE
/dev/sdb        lvm2 a--  9.09t 9.09t   1.25m
```

You can also set the dalign option in the PV section.

- vgs: This is an optional variable. This variable replaces the [vgs] section in gdeploy 1.x. vgs variable lists the names to be used while creating volume groups. The number of VG names should match the number of devices or should be left blank. gdeploy will generate names for the VGs. For example:

  ```
  [backend-setup]
  devices=sda,sdb,sdc
  vgs=custom_vg1,custom_vg2,custom_vg3
  ```

  A pattern can be provided for the vgs like custom_vg{1..3}, this will create three vgs.

  ```
  [backend-setup]
  devices=sda,sdb,sdc
  vgs=custom_vg{1..3}
  ```

- pools: This is an optional variable. The variable replaces the [pools] section in gdeploy 1.x. pools lists the thin pool names for the volume.

  ```
  [backend-setup]
  devices=sda,sdb,sdc
  vgs=custom_vg1,custom_vg2,custom_vg3
  pools=custom_pool1,custom_pool2,custom_pool3
  ```

  Similar to vg, pattern can be provided for thin pool names. For example custom_pool{1..3}

- lvs: This is an optional variable. This variable replaces the [lvs] section in gdeploy 1.x. lvs lists the logical volume name for the volume.

```
[backend-setup]
devices=sda,sdb,sdc
vgs=custom_vg1,custom_vg2,custom_vg3
pools=custom_pool1,custom_pool2,custom_pool3
lvs=custom_lv1,custom_lv2,custom_lv3
```

Patterns for LV can be provided similar to vg. For example custom_lv{1..3}.

- mountpoints: This variable deprecates the [mountpoints] section in gdeploy 1.x. Mountpoints lists the mount points where the logical volumes should be mounted. Number of mount points should be equal to the number of logical volumes. For example:

```
[backend-setup]
devices=sda,sdb,sdc
vgs=custom_vg1,custom_vg2,custom_vg3
pools=custom_pool1,custom_pool2,custom_pool3
lvs=custom_lv1,custom_lv2,custom_lv3
mountpoints=/gluster/data1,/gluster/data2,/gluster/data3
```

- ssd - This variable is set if caching has to be added. For example, the backed setup with ssd for caching should be:

```
[backend-setup]
ssd=sdc
vgs=RHS_vg1
datalv=lv_data
cachedatalv=lv_cachedata:1G
cachemetalv=lv_cachemeta:230G
```

Note

Specifying the name of the data LV is necessary while adding SSD. Make sure the datalv is created already. Otherwise ensure to create it in one of the earlier `backend-setup’ sections.

PV

Available in gdeploy 2.0. If the user needs to have more control over setting up the backend, and does not want to use backend-setup section, then pv, vg, and lv modules are to be used. The pv module supports the following variables.

- action: Mandatory. Supports two values, 'create' and 'resize'

  Example: Creating physical volumes

```
[pv]
action=create
devices=vdb,vdc,vdd
```

  Example: Creating physical volumes on a specific host

```
[pv:10.0.5.2]
action=create
devices=vdb,vdc,vdd
```

- devices: Mandatory. The list of devices to use for pv creation.

- expand: Used when **`action=resize`**.

  Example: Expanding an already created pv

  ```
  [pv]
  action=resize
  devices=vdb
  expand=yes
  ```

- shrink: Used when **`action=resize`**.

  Example: Shrinking an already created pv

  ```
  [pv]
  action=resize
  devices=vdb
  shrink=100G
  ```

- dalign:

  The Logical Volume Manager can use a portion of the physical volume for storing its metadata while the rest is used as the data portion. Align the I/O at the Logical Volume Manager (LVM) layer using the dalign option while creating the physical volume. For example:

  ```
  [pv]
  action=create
  devices=sdb,sdc,sdd,sde
  dalign=256k
  ```

  For JBOD, use an alignment value of 256K. For hardware RAID, the alignment value should be obtained by multiplying the RAID stripe unit size with the number of data disks. If 12 disks are used in a RAID 6 configuration, the number of data disks is 10; on the other hand, if 12 disks are used in a RAID 10 configuration, the number of data disks is 6.

  The following example is appropriate for 12 disks in a RAID 6 configuration with a stripe unit size of 128 KiB:

  ```
  [pv]
  action=create
  devices=sdb,sdc,sdd,sde
  dalign=1280k
  ```

  The following example is appropriate for 12 disks in a RAID 10 configuration with a stripe unit size of 256 KiB:

  ```
  [pv]
  action=create
  devices=sdb,sdc,sdd,sde
  dalign=1536k
  ```

To view the previously configured physical volume settings for the dalign option, run the **pvs -o +pe_start *device*** command. For example:

```
# pvs -o +pe_start disk
PV         VG    Fmt  Attr PSize PFree 1st PE
/dev/sdb         lvm2 a--   9.09t 9.09t   1.25m
```

You can also set the dalign option in the backend-setup section.

- ### VG

Available in gdeploy 2.0. This module is used to create and extend volume groups. The vg module supports the following variables.

- action - Action can be one of create or extend.
- pvname - PVs to use to create the volume. For more than one PV use comma separated values.
- vgname - The name of the vg. If no name is provided GLUSTER_vg will be used as default name.
- one-to-one - If set to yes, one-to-one mapping will be done between pv and vg.

If action is set to extend, the vg will be extended to include pv provided.

Example1: Create a vg named images_vg with two PVs

```
[vg]
action=create
vgname=images_vg
pvname=sdb,sdc
```

Example2: Create two vgs named rhgs_vg1 and rhgs_vg2 with two PVs

```
[vg]
action=create
vgname=rhgs_vg
pvname=sdb,sdc
one-to-one=yes
```

Example3: Extend an existing vg with the given disk.

```
[vg]
action=extend
vgname=rhgs_images
pvname=sdc
```

- ### LV

Available in gdeploy 2.0. This module is used to create, setup-cache, and convert logical volumes. The lv module supports the following variables:

action - The action variable allows three values `create’, `setup-cache’, `convert’, and `change’. If the action is 'create', the following options are supported:

- lvname: The name of the logical volume, this is an optional field. Default is GLUSTER_lv

- poolname - Name of the thinpool volume name, this is an optional field. Default is GLUSTER_pool

- lvtype - Type of the logical volume to be created, allowed values are `thin' and `thick'. This is an optional field, default is thick.

- size - Size of the logical volume volume. Default is to take all available space on the vg.

- extent - Extent size, default is 100%FREE

- force - Force lv create, do not ask any questions. Allowed values `yes', `no'. This is an optional field, default is yes.

- vgname - Name of the volume group to use.

- pvname - Name of the physical volume to use.

- chunksize - The size of the chunk unit used for snapshots, cache pools, and thin pools. By default this is specified in kilobytes. For RAID 5 and 6 volumes, gdeploy calculates the default chunksize by multiplying the stripe size and the disk count. For RAID 10, the default chunksize is 256 KB. See Section 21.2, "Brick Configuration" for details.

  Warning

  Red Hat recommends using at least the default chunksize. If the chunksize is too small and your volume runs out of space for metadata, the volume is unable to create data. This includes the data required to increase the size of the metadata pool or to migrate data away from a volume that has run out of metadata space. Red Hat recommends monitoring your logical volumes to ensure that they are expanded or more storage created before metadata volumes become completely full.

- poolmetadatasize - Sets the size of pool's metadata logical volume. Allocate the maximum chunk size (16 GiB) if possible. If you allocate less than the maximum, allocate at least 0.5% of the pool size to ensure that you do not run out of metadata space.

  Warning

  If your metadata pool runs out of space, you cannot create data. This includes the data required to increase the size of the metadata pool or to migrate data away from a volume that has run out of metadata space. Monitor your metadata pool using the **`lvs -o+metadata_percent`** command and ensure that it does not run out of space.

- virtualsize - Creates a thinly provisioned device or a sparse device of the given size

- mkfs - Creates a filesystem of the given type. Default is to use xfs.

- mkfs-opts - mkfs options.

- mount - Mount the logical volume.

If the action is setup-cache, the below options are supported:

- ssd - Name of the ssd device. For example sda/vda/ … to setup cache.

- vgname - Name of the volume group.

- poolname - Name of the pool.
- cache_meta_lv - Due to requirements from dm-cache (the kernel driver), LVM further splits the cache pool LV into two devices - the cache data LV and cache metadata LV. Provide the cache_meta_lv name here.
- cache_meta_lvsize - Size of the cache meta lv.
- cache_lv - Name of the cache data lv.
- cache_lvsize - Size of the cache data.
- force - Force

If the action is convert, the below options are supported:

- lvtype - type of the lv, available options are thin and thick
- force - Force the lvconvert, default is yes.
- vgname - Name of the volume group.
- poolmetadata - Specifies cache or thin pool metadata logical volume.
- cachemode - Allowed values writeback, writethrough. Default is writethrough.
- cachepool - This argument is necessary when converting a logical volume to a cache LV. Name of the cachepool.
- lvname - Name of the logical volume.
- chunksize - The size of the chunk unit used for snapshots, cache pools, and thin pools. By default this is specified in kilobytes. For RAID 5 and 6 volumes, gdeploy calculates the default chunksize by multiplying the stripe size and the disk count. For RAID 10, the default chunksize is 256 KB. See Section 21.2, “Brick Configuration” for details.

  Warning

  Red Hat recommends using at least the default chunksize. If the chunksize is too small and your volume runs out of space for metadata, the volume is unable to create data. Red Hat recommends monitoring your logical volumes to ensure that they are expanded or more storage created before metadata volumes become completely full.

- poolmetadataspare - Controls creation and maintanence of pool metadata spare logical volume that will be used for automated pool recovery.
- thinpool - Specifies or converts logical volume into a thin pool's data volume. Volume’s name or path has to be given.

If the action is change, the below options are supported:

- lvname - Name of the logical volume.
- vgname - Name of the volume group.
- zero - Set zeroing mode for thin pool.

Example 1: Create a thin LV

```
[lv]
action=create
vgname=RHGS_vg1
poolname=lvthinpool
lvtype=thinpool
poolmetadatasize=200MB
chunksize=1024k
size=30GB
```

Example 2: Create a thick LV

```
[lv]
action=create
vgname=RHGS_vg1
lvname=engine_lv
lvtype=thick
size=10GB
mount=/rhgs/brick1
```

If there are more than one LVs, then the LVs can be created by numbering the LV sections, like [lv1], [lv2] …

RH-subscription

Available in gdeploy 2.0. This module is used to subscribe, unsubscribe, attach, enable repos etc. The RH-subscription module allows the following variables:

This module is used to subscribe, unsubscribe, attach, enable repos etc. The RH-subscription module allows the following variables:

If the action is **register**, the following options are supported:

- username/activationkey: Username or activationkey.
- password/activationkey: Password or activation key
- auto-attach: true/false
- pool: Name of the pool.
- repos: Repos to subscribe to.
- disable-repos: Repo names to disable. Leaving this option blank will disable all the repos.
- ignore_register_errors: If set to no, gdeploy will exit if system registration fails.
- If the action is **attach-pool** the following options are supported:

  pool - Pool name to be attached.

  ignore_attach_pool_errors - If set to no, gdeploy fails if attach-pool fails.
- If the action is **enable-repos** the following options are supported:

  repos - List of comma separated repos that are to be subscribed to.

  ignore_enable_errors - If set to no, gdeploy fails if enable-repos fail.
- If the action is **disable-repos** the following options are supported:

repos - List of comma separated repos that are to be subscribed to.

ignore_disable_errors - If set to no, gdeploy fails if disable-repos fail

- If the action is **unregister** the systems will be unregistered.

  ignore_unregister_errors - If set to no, gdeploy fails if unregistering fails.

Example 1: Subscribe to Red Hat Subscription network:

```
[RH-subscription1]
action=register
username=qa@redhat.com
password=<passwd>
pool=<pool>
ignore_register_errors=no
```

Example 2: Disable all the repos:

```
[RH-subscription2]
action=disable-repos
repos=
```

Example 3: Enable a few repos

```
[RH-subscription3]
action=enable-repos
repos=rhel-7-server-rpms,rh-gluster-3-for-rhel-7-server-rpms,rhel-7-
server-rhev-mgmt-agent-rpms
ignore_enable_errors=no
```

yum

Available in gdeploy 2.0. This module is used to install or remove rpm packages, with the yum module we can add repos as well during the install time.

The action variable allows two values `install’ and `remove’.

If the action is install the following options are supported:

- packages - Comma separated list of packages that are to be installed.
- repos - The repositories to be added.
- gpgcheck - yes/no values have to be provided.
- update - Whether yum update has to be initiated.

If the action is remove then only one option has to be provided:

- remove - The comma separated list of packages to be removed.

For example

```
[yum1]
action=install
gpgcheck=no
```

```
# Repos should be an url; eg: http://repo-pointing-glusterfs-builds
repos=<glusterfs.repo>,<vdsm.repo>
packages=vdsm,vdsm-gluster,ovirt-hosted-engine-setup,screen,gluster-
nagios-addons,xauth
update=yes
```

Install a package on a particular host.

```
[yum2:host1]
action=install
gpgcheck=no
packages=rhevm-appliance
```

shell

Available in gdeploy 2.0. This module allows user to run shell commands on the remote nodes.

Currently shell provides a single action variable with value execute. And a command variable with any valid shell command as value.

The below command will execute vdsm-tool on all the nodes.

```
[shell]
action=execute
command=vdsm-tool configure --force
```

update-file

Available in gdeploy 2.0. update-file module allows users to copy a file, edit a line in a file, or add new lines to a file. action variable can be any of copy, edit, or add.

When the action variable is set to copy, the following variables are supported.

- src - The source path of the file to be copied from.
- dest - The destination path on the remote machine to where the file is to be copied to.

When the action variable is set to edit, the following variables are supported.

- dest - The destination file name which has to be edited.
- replace - A regular expression, which will match a line that will be replaced.
- line - Text that has to be replaced.

When the action variable is set to add, the following variables are supported.

- dest - File on the remote machine to which a line has to be added.
- line - Line which has to be added to the file. Line will be added towards the end of the file.

Example 1: Copy a file to a remote machine.

```
[update-file]
action=copy
src=/tmp/foo.cfg
dest=/etc/nagios/nrpe.cfg
```

Example 2: Edit a line in the remote machine, in the below example lines that have allowed_hosts will be replaced with allowed_hosts=host.redhat.com

```
[update-file]
action=edit
dest=/etc/nagios/nrpe.cfg
replace=allowed_hosts
line=allowed_hosts=host.redhat.com
```

Example 3: Add a line to the end of a file

```
[update-file]
action=add
dest=/etc/ntp.conf
line=server clock.redhat.com iburst
```

service

Available in gdeploy 2.0. The service module allows user to start, stop, restart, reload, enable, or disable a service. The action variable specifies these values.

When action variable is set to any of start, stop, restart, reload, enable, disable the variable servicename specifies which service to start, stop etc.

- service - Name of the service to start, stop etc.

Example: enable and start ntp daemon.

```
[service1]
action=enable
service=ntpd
```

```
[service2]
action=restart
service=ntpd
```

script

Available in gdeploy 2.0. script module enables user to execute a script/binary on the remote machine. action variable is set to execute. Allows user to specify two variables file and args.

- file - An executable on the local machine.

- args - Arguments to the above program.

Example: Execute script disable-multipath.sh on all the remote nodes listed in `hosts’ section.

```
[script]
action=execute
file=/usr/share/ansible/gdeploy/scripts/disable-multipath.sh
```

firewalld

Available in gdeploy 2.0. firewalld module allows the user to manipulate firewall rules. action variable supports two values `add’ and `delete’. Both add and delete support the following variables:

- ports/services - The ports or services to add to firewall.
- permanent - Whether to make the entry permanent. Allowed values are true/false
- zone - Default zone is public

For example:

```
[firewalld]
action=add
ports=111/tcp,2049/tcp,54321/tcp,5900/tcp,5900-6923/tcp,5666/tcp,16514/tcp
services=glusterfs
```

5.1.8. Deploying NFS Ganesha using gdeploy

gdeploy supports the deployment and configuration of NFS Ganesha on Red Hat Gluster Storage 3.4, from gdeploy version 2.0.1.

NFS-Ganesha is a user space file server for the NFS protocol. For more information about NFS-Ganesha see https://access.redhat.com/documentation/en-us/red_hat_gluster_storage/3.4/html-single/administration_guide/#nfs_ganesha

5.1.8.1. Prerequisites

Ensure that the following prerequisites are met:

Subscribing to Subscription Manager

You must subscribe to subscription manager and obtain the NFS Ganesha packages before continuing further.

Add the following details to the configuration file to subscribe to subscription manager:

```
[RH-subscription1]
action=register
username=<user>@redhat.com
password=<password>
pool=<pool-id>
```

Execute the following command to run the configuration file:

```
# gdeploy -c <config_file_name>
```

Enabling Repos

To enable the required repos, add the following details in the configuration file:

```
[RH-subscription2]
action=enable-repos
repos=rhel-7-server-rpms,rh-gluster-3-for-rhel-7-server-rpms,rh-gluster-3-
nfs-for-rhel-7-server-rpms,rhel-ha-for-rhel-7-server-rpms,rhel-7-server-
ansible-2-rpms
```

Execute the following command to run the configuration file:

```
# gdeploy -c <config_file_name>
```

Enabling Firewall Ports

To enable the firewall ports, add the following details in the configuration file:

```
[firewalld]
action=add
ports=111/tcp,2049/tcp,54321/tcp,5900/tcp,5900-6923/tcp,5666/tcp,16514/tcp
services=glusterfs,nlm,nfs,rpc-bind,high-availability,mountd,rquota
```

> **Note**
>
> To ensure NFS client UDP mount does not fail, ensure to add port 2049/udp in [firewalld] section of gdeploy.

Execute the following command to run the configuration file:

```
# gdeploy -c <config_file_name>
```

Installing the Required Package:

To install the required package, add the following details in the configuration file

```
[yum]
action=install
repolist=
gpgcheck=no
update=no
packages=glusterfs-ganesha
```

Execute the following command to run the configuration file:

```
# gdeploy -c <config_file_name>
```

5.1.8.2. Supported Actions

The NFS Ganesha module in gdeploy allows the user to perform the following actions:

- Creating a Cluster
- Destroying a Cluster
- Adding a Node
- Deleting a Node
- Exporting a Volume
- Unexporting a Volume
- Refreshing NFS Ganesha Configuration

Creating a Cluster

This action creates a fresh NFS-Ganesha setup on a given volume. For this action the nfs-ganesha in the configuration file section supports the following variables:

- *ha-name*: This is an optional variable. By default it is ganesha-ha-360.
- *cluster-nodes*: This is a required argument. This variable expects comma separated values of cluster node names, which is used to form the cluster.
- *vip*: This is a required argument. This variable expects comma separated list of ip addresses. These will be the virtual ip addresses.
- *volname*: This is an optional variable if the configuration contains the [volume] section

For example: To create a NFS-Ganesha cluster add the following details in the configuration file:

```
[hosts]
host-1.example.com
host-2.example.com

[backend-setup]
devices=/dev/vdb
vgs=vg1
pools=pool1
lvs=lv1
mountpoints=/mnt/brick

[firewalld]
action=add
ports=111/tcp,2049/tcp,54321/tcp,5900/tcp,5900-
6923/tcp,5666/tcp,16514/tcp,662/tcp,662/udp
services=glusterfs,nlm,nfs,rpc-bind,high-availability,mountd,rquota

[volume]
action=create
volname=ganesha
transport=tcp
replica_count=2
force=yes

#Creating a high availability cluster and exporting the volume
[nfs-ganesha]
action=create-cluster
ha-name=ganesha-ha-360
cluster-nodes=host-1.example.com,host-2.example.com
vip=10.70.44.121,10.70.44.122
volname=ganesha
ignore_ganesha_errors=no
```

In the above example, it is assumed that the required packages are installed, a volume is created and NFS-Ganesha is enabled on it.

Execute the configuration using the following command:

```
# gdeploy -c <config_file_name>
```

Destroying a Cluster

The action, destroy-cluster cluster disables NFS Ganesha. It allows one variable, **`cluster-nodes`**.

For example: To destroy a NFS-Ganesha cluster add the following details in the configuration file:

```
[hosts]
host-1.example.com
host-2.example.com

# To destroy the high availability cluster

[nfs-ganesha]
action=destroy-cluster
cluster-nodes=host-1.example.com,host-2.example.com
```

Execute the configuration using the following command:

```
# gdeploy -c <config_file_name>
```

Adding a Node

The add-node action allows three variables:

- **nodes**: Accepts a list of comma separated hostnames that have to be added to the cluster
- **`vip`**: Accepts a list of comma separated ip addresses.
- **`cluster_nodes`**: Accepts a list of comma separated nodes of the NFS Ganesha cluster.

For example, to add a node, add the following details to the configuration file:

```
[hosts]
host-1.example.com
host-2.example.com
host-3.example.com

[peer]
action=probe

[clients]
action=mount
volname=gluster_shared_storage
hosts=host-3.example.com
fstype=glusterfs
client_mount_points=/var/run/gluster/shared_storage/

[nfs-ganesha]
action=add-node
nodes=host-3.example.com
cluster_nodes=host-1.example.com,host-2.example.com
vip=10.0.0.33
```

Execute the configuration using the following command:

```
# gdeploy -c <config_file_name>
```

Deleting a Node

The **delete-node** action takes one variable, ***nodes***, which specifies the node or nodes to delete from the NFS Ganesha cluster in a comma delimited list.

For example:

```
[hosts]
host-1.example.com
host-2.example.com
host-3.example.com
host-4.example.com

[nfs-ganesha]
action=delete-node
nodes=host-2.example.com
```

Exporting a Volume

This action exports a volume. export-volume action supports one variable, **volname**.

For example, to export a volume, add the following details to the configuration file:

```
[hosts]
host-1.example.com
host-2.example.com

[nfs-ganesha]
action=export-volume
volname=ganesha
```

Execute the configuration using the following command:

```
# gdeploy -c <config_file_name>
```

Unexporting a Volume:

This action unexports a volume. unexport-volume action supports one variable, **volname**.

For example, to unexport a volume, add the following details to the configuration file:

```
[hosts]
host-1.example.com
host-2.example.com

[nfs-ganesha]
action=unexport-volume
volname=ganesha
```

Execute the configuration using the following command:

```
# gdeploy -c <config_file_name>
```

Refreshing NFS Ganesha Configuration

Refreshing NFS Ganesha Configuration

This action will add/delete or add a config block to the configuration file and runs **refresh-config** on the cluster.

The action **refresh-config** supports the following variables:

- del-config-lines
- block-name
- volname
- ha-conf-dir
- update_config_lines

Example 1 - To add a client block and run refresh-config add the following details to the configuration file:

> **Note**
>
> **refresh-config** with client block has few limitations:
>
> - Works for only one client
> - User cannot delete a line from a config block

```
[hosts]
host1-example.com
host2-example.com

[nfs-ganesha]
action=refresh-config
# Default block name is `client'
block-name=client
config-block=clients = 10.0.0.1;|allow_root_access = true;|access_type =
"RO";|Protocols = "2", "3";|anonymous_uid = 1440;|anonymous_gid = 72;
volname=ganesha
```

Execute the configuration using the following command:

```
# gdeploy -c <config_file_name>
```

Example 2 - To delete a line and run refresh-config add the following details to the configuration file:

```
[hosts]
host1-example.com
host2-example.com


[nfs-ganesha]
action=refresh-config
del-config-lines=client
volname=ganesha
```

Execute the configuration using the following command:

```
# gdeploy -c <config_file_name>
```

Example 3 - To run refresh-config on a volume add the following details to the configuration file:

```
[hosts]
host1-example.com
host2-example.com


[nfs-ganesha]
action=refresh-config
volname=ganesha
```

Execute the configuration using the following command:

```
# gdeploy -c <config_file_name>
```

Example 4 - To modify a line and run refresh-config add the following details to the configuration file:

```
[hosts]
host1-example.com
host2-example.com


[nfs-ganesha]
action=refresh-config
update_config_lines=Access_type = "RO";
#update_config_lines=Protocols = "4";
#update_config_lines=clients = 10.0.0.1;
volname=ganesha
```

Execute the configuration using the following command:

```
# gdeploy -c <config_file_name>
```

5.1.9. Deploying Samba / CTDB using gdeploy

The Server Message Block (SMB) protocol can be used to access Red Hat Gluster Storage volumes by exporting directories in GlusterFS volumes as SMB shares on the server. In Red Hat Gluster Storage, Samba is used to share volumes through SMB protocol.

5.1.9.1. Prerequisites

Ensure that the following prerequisites are met:

Subscribing to Subscription Manager

You must subscribe to subscription manager and obtain the Samba packages before continuing further.

Add the following details to the configuration file to subscribe to subscription manager:

```
[RH-subscription1]
action=register
username=<user>@redhat.com
password=<password>
pool=<pool-id>
```

Execute the following command to run the configuration file:

```
# gdeploy -c <config_file_name>
```

Enabling Repos

To enable the required repos, add the following details in the configuration file:

```
[RH-subscription2]
action=enable-repos
repos=rhel-7-server-rpms,rh-gluster-3-for-rhel-7-server-rpms,rh-gluster-3-
samba-for-rhel-7-server-rpms,rhel-7-server-ansible-2-rpms
```

Execute the following command to run the configuration file:

```
# gdeploy -c <config_file_name>
```

Enabling Firewall Ports

To enable the firewall ports, add the following details in the configuration file:

```
[firewalld]
action=add
ports=54321/tcp,5900/tcp,5900-6923/tcp,5666/tcp,4379/tcp
services=glusterfs,samba,high-availability
```

Execute the following command to run the configuration file:

```
# gdeploy -c <config_file_name>
```

Installing the Required Package:

To install the required package, add the following details in the configuration file

```
[yum]
action=install
repolist=
gpgcheck=no
update=no
packages=samba,samba-client,glusterfs-server,ctdb
```

Execute the following command to run the configuration file:

```
# gdeploy -c <config_file_name>
```

5.1.9.2. Setting up Samba

Samba can be enabled in two ways:

- Enabling Samba on an existing volume
- Enabling Samba while creating a volume

Enabling Samba on an existing volume

If a Red Hat Gluster Storage volume is already present, then the user has to mention the action as **smb-setup** in the volume section. It is necessary to mention all the hosts that are in the cluster, as gdeploy updates the glusterd configuration files on each of the hosts.

For example, to enable Samba on an existing volume, add the following details to the configuration file:

```
[hosts]
10.70.37.192
10.70.37.88

[volume]
action=smb-setup
volname=samba1
force=yes
smb_username=smbuser
smb_mountpoint=/mnt/smb
```

Note

Ensure that the hosts are not part of the CTDB cluster.

Execute the configuration using the following command:

```
# gdeploy -c <config_file_name>
```

Enabling Samba while creating a Volume

If Samba has be set up while creating a volume, the a variable **smb** has to be set to yes in the configuration file.

For example, to enable Samba while creating a volume, add the following details to the configuration file:

```
[hosts]
10.70.37.192
10.70.37.88

[backend-setup]
devices=/dev/vdb
vgs=vg1
pools=pool1
lvs=lv1
mountpoints=/mnt/brick

[volume]
action=create
volname=samba1
```

```
smb=yes
force=yes
smb_username=smbuser
smb_mountpoint=/mnt/smb
```

Execute the configuration using the following command:

```
# gdeploy -c <config_file_name>
```

Note

In both the cases of enabling Samba, **smb_username** and **smb_mountpoint** are necessary if samba has to be setup with the acls set correctly.

5.1.9.3. Setting up CTDB

Using CTDB requires setting up a separate volume in order to protect the CTDB lock file. Red Hat recommends a replicated volume where the replica count is equal to the number of servers being used as Samba servers.

The following configuration file sets up a CTDB volume across two hosts that are also Samba servers.

```
[hosts]
10.70.37.192
10.70.37.88

[volume]
action=create
volname=ctdb
transport=tcp
replica_count=2
force=yes

[ctdb]
action=setup
public_address=10.70.37.6/24 eth0,10.70.37.8/24 eth0
volname=ctdb
```

You can configure the CTDB cluster to use separate IP addresses by using the ***ctdb_nodes*** parameter, as shown in the following example.

```
[hosts]
10.70.37.192
10.70.37.88

[volume]
action=create
volname=ctdb
transport=tcp
replica_count=2
force=yes
```

```
[ctdb]
action=setup
public_address=10.70.37.6/24 eth0,10.70.37.8/24 eth0
ctdb_nodes=192.168.1.1,192.168.2.5
volname=ctdb
```

Execute the configuration using the following command:

```
# gdeploy -c <config_file_name>
```

5.1.10. Enabling SSL on a Volume

You can create volumes with SSL enabled, or enable SSL on an exisiting volumes using gdeploy (v2.0.1 onwards). This section explains how the configuration files should be written for gdeploy to enable SSL.

5.1.10.1. Creating a Volume and Enabling SSL

To create a volume and enable SSL on it, add the following details to the configuration file:

```
[hosts]
10.70.37.147
10.70.37.47

[backend-setup]
devices=/dev/vdb
vgs=vg1
pools=pool1
lvs=lv1
mountpoints=/mnt/brick

[volume]
action=create
volname=vol1
transport=tcp
replica_count=2
force=yes
enable_ssl=yes
ssl_clients=10.70.37.107,10.70.37.173
brick_dirs=/data/1

[clients]
action=mount
hosts=10.70.37.173,10.70.37.107
volname=vol1
fstype=glusterfs
client_mount_points=/mnt/data
```

In the above example, a volume named vol1 is created and SSL is enabled on it. gdeploy creates self signed certificates.

After adding the details to the configuration file, execute the following command to run the configuration file:

```
# gdeploy -c <config_file_name>
```

5.1.10.2. Enabling SSL on an Existing Volume:

To enable SSL on an existing volume, add the following details to the configuration file:

```
[hosts]
10.70.37.147
10.70.37.47

# It is important for the clients to be unmounted before setting up SSL
[clients1]
action=unmount
hosts=10.70.37.173,10.70.37.107
client_mount_points=/mnt/data

[volume]
action=enable-ssl
volname=vol2
ssl_clients=10.70.37.107,10.70.37.173

[clients2]
action=mount
hosts=10.70.37.173,10.70.37.107
volname=vol2
fstype=glusterfs
client_mount_points=/mnt/data
```

After adding the details to the configuration file, execute the following command to run the configuration file:

```
# gdeploy -c <config_file_name>
```

5.1.11. Limiting Gluster Resources

When Red Hat Gluster Storage is deployed on the same machine as other resource intensive software and services, it can be useful to limit the resources that glusterd attempts to use to avoid resource contention between processes.

To limit the resources available to glusterd on a Red Hat Enterpise Linux 7 based installation of Red Hat Gluster Storage 3.2 or higher, define **`slice_setup=yes`** when you start the glusterd service. This applies a set of resource limitations for the glusterd service and all of its child processes.

```
[hosts]
192.168.100.101
192.168.100.102
192.168.100.103

[service]
action=start
service=glusterd
slice_setup=yes
```

The resource limitations set cannot be customized using gdeploy, but they can be manually modified outside the scope of gdeploy, for example, by using **`systemctl`**.

If you use a version of Red Hat Gluster Storage that is based on Red Hat Enterprise Linux 6, you cannot set up resource management using gdeploy. See Chapter 20, *Managing Resource Usage* for details.

For more information about resource management, see the Red Hat Enterprise Linux *Resource Management Guide*:

- For Red Hat Enterprise Linux 7: https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html-single/Resource_Management_Guide/index.html#sec-What_are_Control_Groups
- For Red Hat Enterprise Linux 6: https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Resource_Management_Guide/ch-Using_Control_Groups.html

5.1.12. Gdeploy log files

Because gdeploy is usually run by non-privileged users, by default, gdeploy log files are written to **/home/*username*/.gdeploy/logs/gdeploy.log** instead of the **/var/log** directory.

You can change the log location by setting a different location as the value of the ***GDEPLOY_LOGFILE*** environment variable. For example, to set the gdeploy log location to **/var/log/gdeploy/gdeploy.log** for this session, run the following command:

```
$ export GDEPLOY_LOGFILE=/var/log/gdeploy/gdeploy.log
```

To persistently set this as the default log location for this user, add the same command as a separate line in the **/home/*username*/.bash_profile** file for that user.

5.2. Managing Volumes using Heketi

Heketi provides a RESTful management interface which can be used to manage the lifecycle of Red Hat Gluster Storage volumes. With Heketi, cloud services like OpenStack Manila, Kubernetes, and OpenShift can dynamically provision Red Hat Gluster Storage volumes with any of the supported durability types. Heketi will automatically determine the location for bricks across the cluster, making sure to place bricks and its replicas across different failure domains. Heketi also supports any number of Red Hat Gluster Storage clusters, allowing cloud services to provide network file storage without being limited to a single Red Hat Gluster Storage cluster.

With Heketi, the administrator no longer manages or configures bricks, disks, or trusted storage pools. Heketi service will manage all hardware for the administrator, enabling it to allocate storage on demand. Any disks registered with Heketi must be provided in raw format, which will then be managed by it using LVM on the disks provided.

Note

The replica 3 volume type is the default and the only supported volume type that can be created using Heketi.

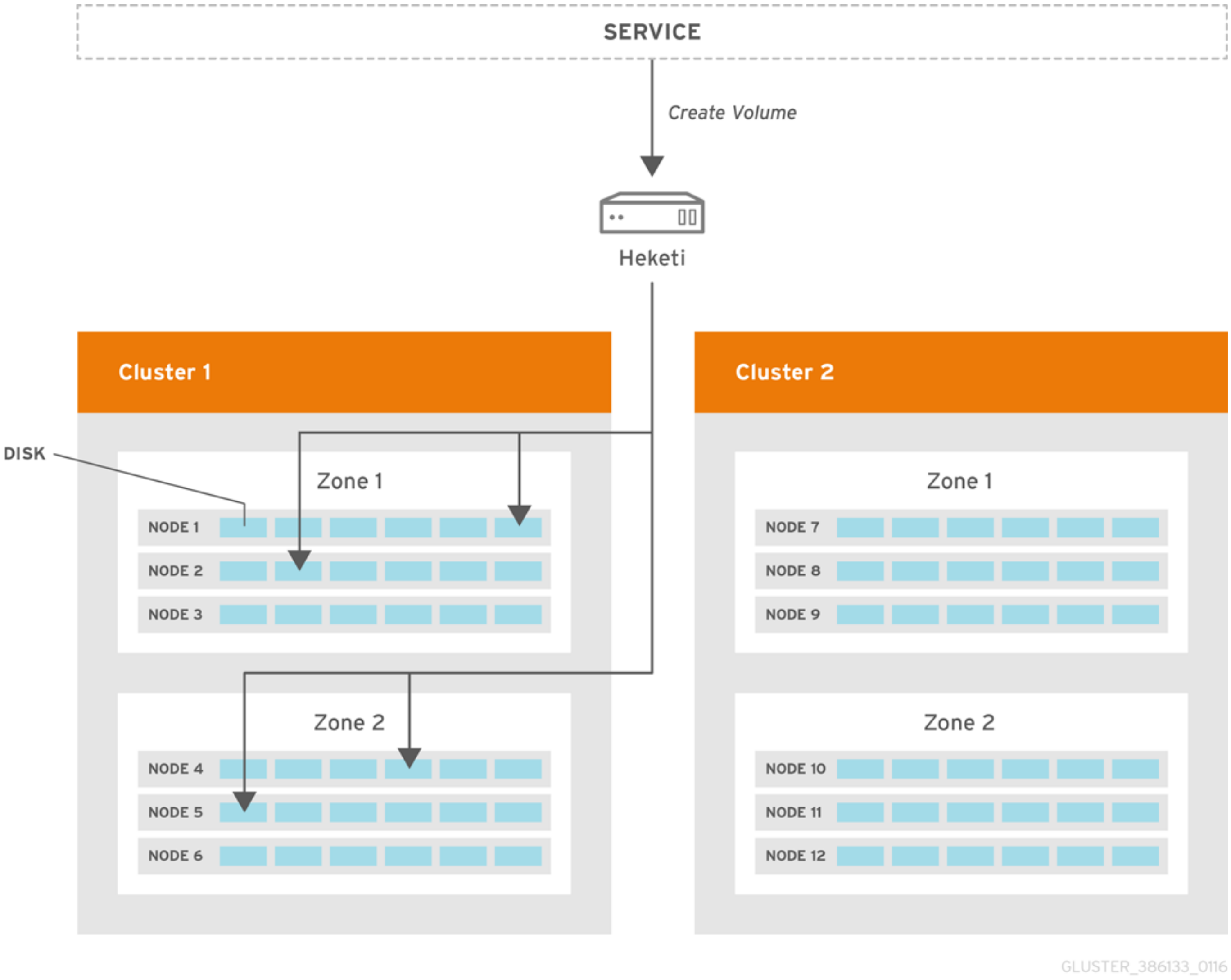


Figure 5.1. Heketi volume creation

A create volume request to Heketi leads it to select bricks spread across 2 zones and 4 nodes. After the volume is created in Red hat Gluster Storage, Heketi provides the volume information to the service that initially made the request.

Heketi can be configured and executed using the CLI or the API. The sections ahead describe configuring Heketi using the CLI.

5.2.1. Prerequisites

Ensure that the following requirements are met:

Configure SSH access

Configure key-based SSH authentication without a password for the Heketi user. For a non-root user:

- Ensure the user and server specified when copying SSH keys matches the user provided to Heketi in the Heketi configuration file.
- Ensure the user can use **sudo** by disabling **`requiretty`** in the **`/etc/sudoers`** file and adding **`sudo: true`** to the sshexec configuration section in the Heketi configuration file.

Configure the firewall

Ensure that Heketi can accept TCP requests over the port specified in the **heketi.json** file. For example, on Red Hat Enterprise Linux 7 based installations, run the following commands:

```
# firewall-cmd --zone=zone_name --add-port=port/tcp
# firewall-cmd --zone=zone_name --add-port=port/tcp --permanent
```

On Red Hat Enterprise Linux 6 based installations, run the following commands:

```
# iptables -A INPUT -m state --state NEW -m tcp -p tcp --dport port -
j ACCEPT
# service iptables save
```

Start glusterd

After Red Hat Gluster Storage is installed, ensure that the **glusterd** service is started.

Ensure disks are raw format

Disks to be registered with Heketi must be in the raw format.

5.2.2. Installing Heketi

Note

Heketi is supported only on Red Hat Enterprise Linux 7.

After installing Red Hat Gluster Storage 3.4, execute the following command to install the heketi-client:

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

heketi-client has the binary for the heketi command line tool.

Execute the following command to install heketi:

```
# yum install heketi
```

For more information about subscribing to the required channels and installing Red Hat Gluster Storage, see the *Red Hat Gluster Storage Installation Guide*.

5.2.3. Starting the Heketi Server

Before starting the server, ensure that the following prerequisites are met:

- Generate the passphrase-less SSH keys for the nodes which are going to be part of the trusted storage pool by running the following command:

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

- Change the owner and the group permissions for the heketi keys using the following command:

```
# chown heketi:heketi /etc/heketi/heketi_key*
```

- Set up key-based SSH authentication access between Heketi and the Red Hat Gluster Storage servers by running the following command:

  ```
  # ssh-copy-id -i /etc/heketi/heketi_key.pub root@server
  ```

- As a non root user, set up password-less SSH access between Heketi and the Red Hat Gluster Storage servers by running the following command:

  ```
  $ ssh-copy-id -i /etc/heketi/heketi_key.pub user@server
  ```

- > **Note**
  >
  > To run SSH as a non-root, the username mentioned in **`username>@server`** for ssh-copy-id must match with the user name provided to Heketi in the Heketi configuration file below.

- Setup the heketi.json configuration file. The file is located in /etc/heketi/heketi.json. The configuration file has the information required to run the Heketi server. The config file must be in JSON format with the following settings:

  - port: string, Heketi REST service port number
  - use_auth: bool, Enable JWT Authentication
  - jwt: map, JWT Authentication settings
    - admin: map, Settings for the Heketi administrator
      - key: string,
      - user: map, Settings for the Heketi volume requests access user
      - key: string, t
  - glusterfs: map, Red Hat Gluster Storage settings
    - executor: string, Determines the type of command executor to use. Possible values are:
      - mock: Does not send any commands out to servers. Can be used for development and tests
      - ssh: Sends commands to real systems over ssh
    - db: string, Location of Heketi database
    - sshexec: map, SSH configuration
      - keyfile: string, File with private ssh key
      - user: string, SSH user

  Following is an example of the JSON file:

  ```
  {
    "_port_comment": "Heketi Server Port Number",
  ```

```
  "port": "8080",

  "_use_auth": "Enable JWT authorization. Please enable for deployment",
  "use_auth": false,

  "_jwt": "Private keys for access",
  "jwt": {
    "_admin": "Admin has access to all APIs",
    "admin": {
      "key": "My Secret"
    },
    "_user": "User only has access to /volumes endpoint",
    "user": {
      "key": "My Secret"
    }
  },

  "_glusterfs_comment": "GlusterFS Configuration",
  "glusterfs": {
    "_executor_comment": [
      "Execute plugin. Possible choices: mock, ssh",
      "mock: This setting is used for testing and development.",
      "      It will not send commands to any node.",
      "ssh:  This setting will notify Heketi to ssh to the nodes.",
      "      It will need the values in sshexec to be configured.",
      "kubernetes: Communicate with GlusterFS containers over",
      "            Kubernetes exec api."
    ],
    "executor": "ssh",

    "_sshexec_comment": "SSH username and private key file information",
    "sshexec": {
      "keyfile": "path/to/private_key",
      "user": "sshuser",
      "port": "Optional: ssh port.  Default is 22",
      "fstab": "Optional: Specify fstab file on node.  Default is
/etc/fstab",
      "sudo": "Optional: set to true if SSH as a non root user. Default
is false."
    },

    "_kubeexec_comment": "Kubernetes configuration",
    "kubeexec": {
      "host" :"https://kubernetes.host:8443",
      "cert" : "/path/to/crt.file",
      "insecure": false,
      "user": "kubernetes username",
      "password": "password for kubernetes user",
      "namespace": "OpenShift project or Kubernetes namespace",
      "fstab": "Optional: Specify fstab file on node.  Default is
/etc/fstab"
    },

    "_db_comment": "Database file name",
    "db": "/var/lib/heketi/heketi.db",
    "_loglevel_comment": [
      "Set log level. Choices are:",
      "  none, critical, error, warning, info, debug",
      "Default is warning"
    ],
    "loglevel" : "debug"
  }
}
```

> **Note**
>
> The location for the private SSH key that is created must be set in the **`keyfile`** setting of the configuration file, and the key should be readable by the heketi user.

5.2.3.1. Starting the Server

For Red Hat Enterprise Linux 7

1. Enable heketi by executing the following command:

   ```
   # systemctl enable heketi
   ```

2. Start the Heketi server, by executing the following command:

   ```
   # systemctl start heketi
   ```

3. To check the status of the Heketi server, execute the following command:

   ```
   # systemctl status heketi
   ```

4. To check the logs, execute the following command:

   ```
   # journalctl -u heketi
   ```

> **Note**
>
> After Heketi is configured to manage the trusted storage pool, gluster commands should not be run on it, as this will make the heketidb inconsistent, leading to unexpected behaviors with Heketi.

5.2.3.2. Verifying the Configuration

To verify if the server is running, execute the following step:

If Heketi is not setup with authentication, then use curl to verify the configuration:

```
# curl http://<server:port>/hello
```

You can also verify the configuration using the heketi-cli when authentication is enabled:

```
# heketi-cli --server http://<server:port> --user <user> --secret <secret>
cluster list
```

5.2.4. Setting up the Topology

Setting up the topology allows Heketi to determine which nodes, disks, and clusters to use.

5.2.4.1. Prerequisites

You have to determine the node failure domains and clusters of nodes. Failure domains is a value given to a set of nodes which share the same switch, power supply, or anything else that would cause them to fail at the same time. Heketi uses this information to make sure that replicas are created across failure domains, thus providing cloud services volumes which are resilient to both data unavailability and data loss.

You have to determine which nodes would constitute a cluster. Heketi supports multiple Red Hat Gluster Storage clusters, which gives cloud services the option of specifying a set of clusters where a volume must be created. This provides cloud services and administrators the option of creating SSD, SAS, SATA, or any other type of cluster which provide a specific quality of service to users.

Note

Heketi does not have a mechanism today to study and build its database from an existing system. So, a new trusted storage pool has to be configured that can be used by Heketi.

5.2.4.2. Topology Setup

The command line client loads the information about creating a cluster, adding nodes to that cluster, and then adding disks to each one of those nodes.This information is added into the topology file. To load a topology file with heketi-cli, execute the following command:

Note

A sample, formatted topology file (**`topology-sample.json`**) is installed with the ‘heketi-client’ package in the **`/usr/share/heketi/`** directory.

```
# export HEKETI_CLI_SERVER=http://<heketi_server:port>
# heketi-cli topology load --json=<topology_file>
```

Where **`topology_file`** is a file in JSON format describing the clusters, nodes, and disks to add to Heketi. The format of the file is as follows:

clusters: Array of clusters

- Each element on the array is a map which describes the cluster as follows
  - nodes: Array of nodes in a cluster

    Each element on the array is a map which describes the node as follows
    - node: Same as Node Add, except there is no need to supply the cluster ID.

- devices: Name of each disk to be added
- zone: The value represents failure domain on which the node exists.

For example:

1. Topology file:

```
{
    "clusters": [
        {
            "nodes": [
                {
                    "node": {
                        "hostnames": {
                            "manage": [
                                "10.0.0.1"
                            ],
                            "storage": [
                                "10.0.0.1"
                            ]
                        },
                        "zone": 1
                    },
                    "devices": [
                        "/dev/sdb",
                        "/dev/sdc",
                        "/dev/sdd",
                        "/dev/sde",
                        "/dev/sdf",
                        "/dev/sdg",
                        "/dev/sdh",
                        "/dev/sdi"
                    ]
                },
                {
                    "node": {
                        "hostnames": {
                            "manage": [
                                "10.0.0.2"
                            ],
                            "storage": [
                                "10.0.0.2"
                            ]
                        },
                        "zone": 2
                    },
                    "devices": [
                        "/dev/sdb",
                        "/dev/sdc",
                        "/dev/sdd",
                        "/dev/sde",
                        "/dev/sdf",
                        "/dev/sdg",
                        "/dev/sdh",
                        "/dev/sdi"
                    ]
                },

.......
.......
```

2. Load the Heketi JSON file:

```
# heketi-cli topology load --json=topology_libvirt.json
Creating cluster ... ID: a0d9021ad085b30124afbcf8df95ec06
        Creating node 192.168.10.100 ... ID:
b455e763001d7903419c8ddd2f58aea0
                Adding device /dev/vdb ... OK
                Adding device /dev/vdc ... OK
…….
        Creating node 192.168.10.101 ... ID:
4635bc1fe7b1394f9d14827c7372ef54
                Adding device /dev/vdb ... OK
                Adding device /dev/vdc ... OK
………….
```

3. Execute the following command to check the details of a particular node:

```
# heketi-cli node info b455e763001d7903419c8ddd2f58aea0
Node Id: b455e763001d7903419c8ddd2f58aea0
Cluster Id: a0d9021ad085b30124afbcf8df95ec06
Zone: 1
Management Hostname: 192.168.10.100
Storage Hostname: 192.168.10.100
Devices:
Id:0ddba53c70537938f3f06a65a4a7e88b   Name:/dev/vdi            Size
(GiB):499     Used (GiB):0       Free (GiB):499
Id:4fae3aabbaf79d779795824ca6dc433a   Name:/dev/vdg            Size
(GiB):499     Used (GiB):0       Free (GiB):499
…………….
```

4. Execute the following command to check the details of the cluster:

```
# heketi-cli cluster info a0d9021ad085b30124afbcf8df95ec06
Cluster id: a0d9021ad085b30124afbcf8df95ec06
Nodes:
4635bc1fe7b1394f9d14827c7372ef54
802a3bfab2d0295772ea4bd39a97cd5e
b455e763001d7903419c8ddd2f58aea0
ff9eeb735da341f8772d9415166b3f9d
Volumes:
```

5. To check the details of the device, execute the following command:

```
# heketi-cli device info 0ddba53c70537938f3f06a65a4a7e88b
Device Id: 0ddba53c70537938f3f06a65a4a7e88b
Name: /dev/vdi
Size (GiB): 499
Used (GiB): 0
Free (GiB): 499
Bricks:
```

5.2.5. Creating a Volume

After Heketi is set up, you can use the CLI to create a volume.

1. Execute the following command to check the various option for creating a volume:

```
# heketi-cli volume create --size=<size in Gb> [options]
```

2. **For example**: After setting up the topology file with two nodes on one failure domain, and two nodes in another failure domain, create a 100Gb volume using the following command:

```
# heketi-cli volume create --size=100
Name: vol_0729fe8ce9cee6eac9ccf01f84dc88cc
Size: 100
Id: 0729fe8ce9cee6eac9ccf01f84dc88cc
Cluster Id: a0d9021ad085b30124afbcf8df95ec06
Mount: 192.168.10.101:vol_0729fe8ce9cee6eac9ccf01f84dc88cc
Mount Options: backupvolfile-servers=192.168.10.100,192.168.10.102
Durability Type: replicate
Replica: 3
Snapshot: Disabled

Bricks:
Id: 8998961142c1b51ab82d14a4a7f4402d
Path:
/var/lib/heketi/mounts/vg_0ddba53c70537938f3f06a65a4a7e88b/brick_89989
61142c1b51ab82d14a4a7f4402d/brick
Size (GiB): 50
Node: b455e763001d7903419c8ddd2f58aea0
Device: 0ddba53c70537938f3f06a65a4a7e88b
 …………….
```

3. To check the details of the device, execute the following command:

```
# heketi-cli device info 0ddba53c70537938f3f06a65a4a7e88b
Device Id: 0ddba53c70537938f3f06a65a4a7e88b
Name: /dev/vdi
Size (GiB): 499
Used (GiB): 201
Free (GiB): 298
Bricks:
Id:0f1766cc142f1828d13c01e6eed12c74   Size (GiB):50      Path:
/var/lib/heketi/mounts/vg_0ddba53c70537938f3f06a65a4a7e88b/brick_0f176
6cc142f1828d13c01e6eed12c74/brick
Id:5d944c47779864b428faa3edcaac6902   Size (GiB):50      Path:
/var/lib/heketi/mounts/vg_0ddba53c70537938f3f06a65a4a7e88b/brick_5d944
c47779864b428faa3edcaac6902/brick
Id:8998961142c1b51ab82d14a4a7f4402d   Size (GiB):50      Path:
/var/lib/heketi/mounts/vg_0ddba53c70537938f3f06a65a4a7e88b/brick_89989
61142c1b51ab82d14a4a7f4402d/brick
Id:a11e7246bb21b34a157e0e1fd598b3f9   Size (GiB):50      Path:
/var/lib/heketi/mounts/vg_0ddba53c70537938f3f06a65a4a7e88b/brick_a11e7
246bb21b34a157e0e1fd598b3f9/brick
```

5.2.6. Expanding a Volume

Heketi expands a volume size by using add-brick command. The volume id has to be provided to perform volume expansion.

1. Find the volume id using the volume list command.

```
# heketi-cli volume list
Id:9d219903604cabed5ba234f4f04b2270
Cluster:dab7237f6d6d4825fca8b83a0fac24ac
Name:vol_9d219903604cabed5ba234f4f04b2270
Id:a8770efe13a2269a051712905449f1c1
Cluster:dab7237f6d6d4825fca8b83a0fac24ac    Name:user1vol1
```

2. This volume id can be used as input to heketi-cli for expanding the volume.

```
# heketi-cli volume expand --volume <volume_id> --expand-size <size>
```

For example:

```
# heketi-cli volume expand --volume a8770efe13a2269a051712905449f1c1 -
-expand-size 30
Name: user1vol1
Size: 130
Volume Id: a8770efe13a2269a051712905449f1c1
Cluster Id: dab7237f6d6d4825fca8b83a0fac24ac
Mount: 192.168.21.14:user1vol1
Mount Options: backup-volfile-servers=192.168.21.15,192.168.21.16
Block: false
Free Size: 0
Block Volumes: []
Durability Type: replicate
Distributed+Replica: 3
```

5.2.7. Deleting a Volume

To delete a volume, execute the following command:

```
# heketi-cli volume delete <vol_id>
```

For example:

```
$ heketi-cli volume delete 0729fe8ce9cee6eac9ccf01f84dc88cc
Volume 0729fe8ce9cee6eac9ccf01f84dc88cc deleted
```

5.3. About Encrypted Disk

Red Hat Gluster Storage provides the ability to create bricks on encrypted devices to restrict data access. Encrypted bricks can be used to create Red Hat Gluster Storage volumes.

For information on creating encrypted disk, refer to the following product documentation:

- For RHEL 6, see *Disk Encryption* Appendix of the *Red Hat Enterprise Linux 6 Installation Guide*.

- For RHEL 7, see *Encryption* of the *Red Hat Enterprise Linux 7 Security Guide*.

5.4. Formatting and Mounting Bricks

To create a Red Hat Gluster Storage volume, specify the bricks that comprise the volume. After creating the volume, the volume must be started before it can be mounted.

5.4.1. Creating Bricks Manually

Important

- Red Hat supports formatting a Logical Volume using the XFS file system on the bricks.

5.4.1.1. Creating a Thinly Provisioned Logical Volume

1. Create a physical volume(PV) by using the **pvcreate** command.

   ```
   # pvcreate --dataalignment alignment_value device
   ```

   For example:

   ```
   # pvcreate --dataalignment 1280K /dev/sdb
   ```

   Here, **/dev/sdb** is a storage device.

   Use the correct **dataalignment** option based on your device. For more information, see Section 21.2, “Brick Configuration”

   Note

   The device name and the alignment value will vary based on the device you are using.

2. Create a Volume Group (VG) from the PV using the **vgcreate** command:

   ```
   # vgcreate --physicalextentsize alignment_value volgroup device
   ```

   For example:

   ```
   # vgcreate --physicalextentsize 1280K rhs_vg /dev/sdb
   ```

3. Create a thin-pool using the following commands:

   ```
   # lvcreate --thin volgroup/poolname --size pool_sz --chunksize
   chunk_sz --poolmetadatasize metadev_sz --zero n
   ```

   For example:

```
# lvcreate --thin rhs_vg/rhs_pool --size 2T --chunksize 1280K --
poolmetadatasize 16G --zero n
```

Ensure you read Chapter 21, *Tuning for Performance* to select appropriate values for ***chunksize*** and ***poolmetadatasize***.

4. Create a thinly provisioned volume that uses the previously created pool by running the **lvcreate** command with the **--virtualsize** and **--thin** options:

   ```
   # lvcreate --virtualsize size --thin volgroup/poolname --name volname
   ```

   For example:

   ```
   # lvcreate --virtualsize 1G --thin rhs_vg/rhs_pool --name rhs_lv
   ```

   It is recommended that only one LV should be created in a thin pool.

5. Format bricks using the supported XFS configuration, mount the bricks, and verify the bricks are mounted correctly. To enhance the performance of Red Hat Gluster Storage, ensure you read Chapter 21, *Tuning for Performance* before formatting the bricks.

   Important

   Snapshots are not supported on bricks formatted with external log devices. Do not use **-l logdev=device** option with **mkfs.xfs** command for formatting the Red Hat Gluster Storage bricks.

   ```
   # mkfs.xfs -f -i size=512 -n size=8192 -d su=128k,sw=10 device
   ```

   DEVICE is the created thin LV. The inode size is set to 512 bytes to accommodate for the extended attributes used by Red Hat Gluster Storage.

6. Run **# mkdir *mountpoint*** to create a directory to link the brick to.

   ```
   # mkdir /rhgs
   ```

7. Add an entry in **/etc/fstab**:

   ```
   /dev/volgroup/volname /mountpoint  xfs rw,inode64,noatime,nouuid,x-
   systemd.device-timeout=10min  1 2
   ```

   For example:

   ```
   /dev/rhs_vg/rhs_lv /rhgs  xfs rw,inode64,noatime,nouuid,x-
   systemd.device-timeout=10min  1 2
   ```

8. Run **mount */mountpoint*** to mount the brick.

9. Run the **df -h** command to verify the brick is successfully mounted:

   ```
   # df -h
   /dev/rhs_vg/rhs_lv   16G  1.2G   15G   7% /rhgs
   ```

10. If SElinux is enabled, then the SELinux labels that has to be set manually for the bricks created using the following commands:

    ```
    # semanage fcontext -a -t glusterd_brick_t /rhgs/brick1
    # restorecon -Rv /rhgs/brick1
    ```

5.4.2. Using Subdirectory as the Brick for Volume

You can create an XFS file system, mount them and point them as bricks while creating a Red Hat Gluster Storage volume. If the mount point is unavailable, the data is directly written to the root file system in the unmounted directory.

For example, the **/rhgs** directory is the mounted file system and is used as the brick for volume creation. However, for some reason, if the mount point is unavailable, any write continues to happen in the **/rhgs** directory, but now this is under root file system.

To overcome this issue, you can perform the below procedure.

During Red Hat Gluster Storage setup, create an XFS file system and mount it. After mounting, create a subdirectory and use this subdirectory as the brick for volume creation. Here, the XFS file system is mounted as **/bricks**. After the file system is available, create a directory called **/rhgs/brick1** and use it for volume creation. Ensure that no more than one brick is created from a single mount. This approach has the following advantages:

- When the **/rhgs** file system is unavailable, there is no longer**/rhgs/brick1** directory available in the system. Hence, there will be no data loss by writing to a different location.
- This does not require any additional file system for nesting.

Perform the following to use subdirectories as bricks for creating a volume:

1. Create the **brick1** subdirectory in the mounted file system.

   ```
   # mkdir /rhgs/brick1
   ```

   Repeat the above steps on all nodes.

2. Create the Red Hat Gluster Storage volume using the subdirectories as bricks.

   ```
   # gluster volume create distdata01 ad-rhs-srv1:/rhgs/brick1
   ad-rhs-srv2:/rhgs/brick2
   ```

3. Start the Red Hat Gluster Storage volume.

   ```
   # gluster volume start distdata01
   ```

4. Verify the status of the volume.

   ```
   # gluster  volume status distdata01
   ```

Note

If multiple bricks are used from the same server, then ensure the bricks are mounted in the following format. For example:

```
# df -h

/dev/rhs_vg/rhs_lv1   16G  1.2G   15G   7% /rhgs1
/dev/rhs_vg/rhs_lv2   16G  1.2G   15G   7% /rhgs2
```

Create a distribute volume with 2 bricks from each server. For example:

```
# gluster volume create test-volume server1:/rhgs1/brick1
server2:/rhgs1/brick1 server1:/rhgs2/brick2 server2:/rhgs2/brick2
```

5.4.3. Reusing a Brick from a Deleted Volume

Bricks can be reused from deleted volumes, however some steps are required to make the brick reusable.

Brick with a File System Suitable for Reformatting (Optimal Method)

Run `# mkfs.xfs -f -i size=512 device` to reformat the brick to supported requirements, and make it available for immediate reuse in a new volume.

Note

All data will be erased when the brick is reformatted.

File System on a Parent of a Brick Directory

If the file system cannot be reformatted, remove the whole brick directory and create it again.

5.4.4. Cleaning An Unusable Brick

If the file system associated with the brick cannot be reformatted, and the brick directory cannot be removed, perform the following steps:

1. Delete all previously existing data in the brick, including the `.glusterfs` subdirectory.
2. Run `# setfattr -x trusted.glusterfs.volume-id brick` and `# setfattr -x trusted.gfid brick` to remove the attributes from the root of the brick.
3. Run `# getfattr -d -m . brick` to examine the attributes set on the volume. Take note of the attributes.
4. Run `# setfattr -x attribute brick` to remove the attributes relating to the glusterFS file system.

   The `trusted.glusterfs.dht` attribute for a distributed volume is one such example of attributes that need to be removed.

5.5. Creating Distributed Volumes

This type of volume spreads files across the bricks in the volume.

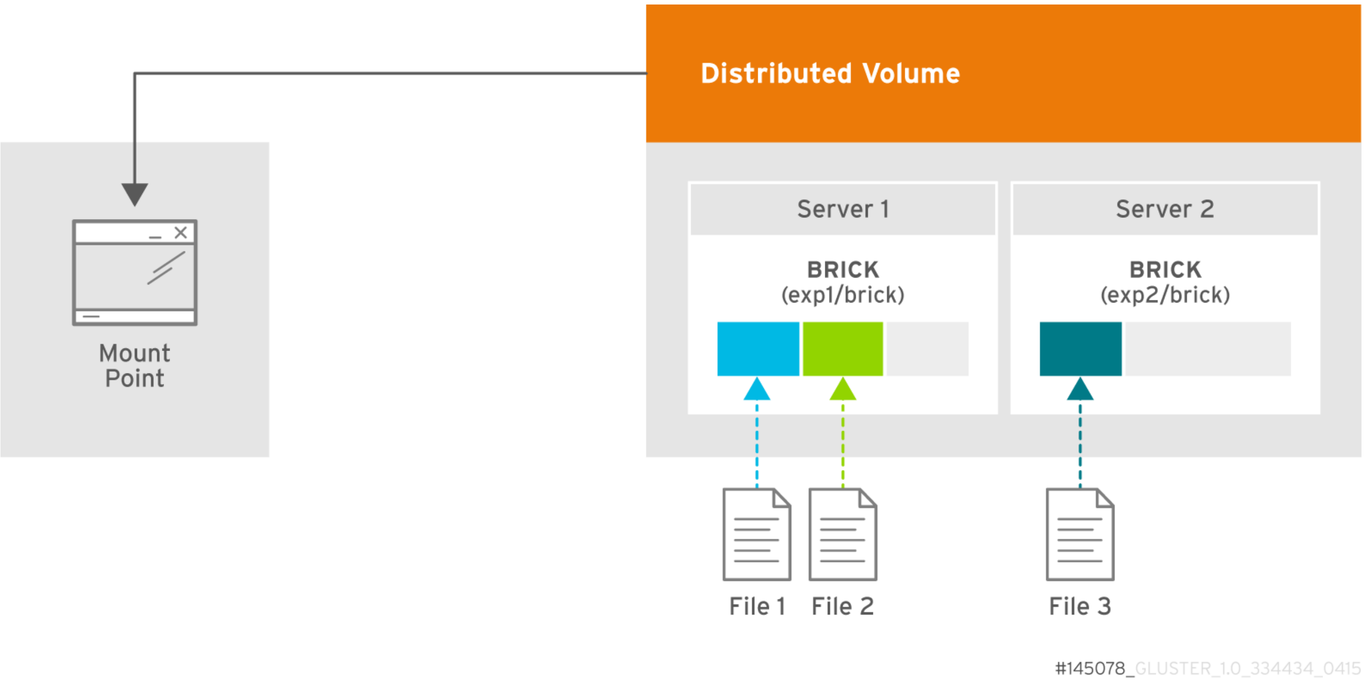


Figure 5.2. Illustration of a Distributed Volume

Warning

Distributed volumes can suffer significant data loss during a disk or server failure because directory contents are spread randomly across the bricks in the volume.

Use distributed volumes where scalable storage and redundancy is either not important, or is provided by other hardware or software layers.

Create a Distributed Volume

Use **`gluster volume create`** command to create different types of volumes, and **`gluster volume info`** command to verify successful volume creation.

Prerequisites

- A trusted storage pool has been created, as described in Section 4.1, “Adding Servers to the Trusted Storage Pool”.
- Understand how to start and stop volumes, as described in Section 5.11, “Starting Volumes”.

1. Run the **`gluster volume create`** command to create the distributed volume.

   The syntax is **`gluster volume create`** ***`NEW-VOLNAME`*** **`[transport tcp | rdma | tcp,rdma]`** ***`NEW-BRICK...`***

   The default value for transport is **`tcp`**. Other options can be passed such as **`auth.allow`** or **`auth.reject`**. See Section 11.1, “Configuring Volume Options” for a full list of parameters.

Red Hat recommends disabling the **performance.client-io-threads** option on distributed volumes, as this option tends to worsen performance. Run the following command to disable **performance.client-io-threads**:

```
# gluster volume set VOLNAME performance.client-io-threads off
```

Example 5.1. Distributed Volume with Two Storage Servers

```
# gluster volume create test-volume server1:/rhgs/brick1
server2:/rhgs/brick1
Creation of test-volume has been successful
Please start the volume to access data.
```

Example 5.2. Distributed Volume over InfiniBand with Four Servers

```
# gluster volume create test-volume transport rdma
server1:/rhgs/brick1 server2:/rhgs/brick1 server3:/rhgs/brick1
server4:/rhgs/brick1
Creation of test-volume has been successful
Please start the volume to access data.
```

2. Run **# gluster volume start *VOLNAME*** to start the volume.

```
# gluster volume start test-volume
Starting test-volume has been successful
```

3. Run **gluster volume info** command to optionally display the volume information.

   The following output is the result of Example 5.1, “Distributed Volume with Two Storage Servers”.

```
# gluster volume info
Volume Name: test-volume
Type: Distribute
Status: Created
Number of Bricks: 2
Transport-type: tcp
Bricks:
Brick1: server1:/rhgs/brick
Brick2: server2:/rhgs/brick
```

5.6. Creating Replicated Volumes

Replicated volume creates copies of files across multiple bricks in the volume. Use replicated volumes in environments where high-availability and high-reliability are critical.

Use **gluster volume create** to create different types of volumes, and **gluster volume info** to verify successful volume creation.

Prerequisites

- A trusted storage pool has been created, as described in Section 4.1, “Adding Servers to the Trusted Storage Pool”.
- Understand how to start and stop volumes, as described in Section 5.11, “Starting Volumes”.

5.6.1. Creating Two-way Replicated Volumes (Deprecated)

Warning

As of Red Hat Gluster Storage 3.4, two-way replication without arbiter bricks is considered deprecated. Existing volumes that use two-way replication without arbiter bricks remain supported for this release. New volumes with this configuration are not supported. Red Hat no longer recommends the use of two-way replication without arbiter bricks, and plans to remove support entirely in future versions of Red Hat Gluster Storage. This change affects both replicated and distributed-replicated volumes that do not use arbiter bricks.

Two-way replication without arbiter bricks is being deprecated because it does not provide adequate protection from split-brain conditions. Even in distributed-replicated configurations, two-way replication cannot ensure that the correct copy of a conflicting file is selected without the use of a tie-breaking node.

While a dummy node can be used as an interim solution for this problem, Red Hat strongly recommends that all volumes that currently use two-way replication without arbiter bricks are migrated to use either arbitrated replication or three-way replication.

Instructions for migrating a two-way replicated volume without arbiter bricks to an arbitrated replicated volume are available in the Red Hat Gluster Storage 3.4 *Administration Guide*: https://access.redhat.com/documentation/en-us/red_hat_gluster_storage/3.4/html-single/administration_guide/#sect-Convert_Rep_to_Arbiter.

Information about three-way replication is available in Section 5.6.2, “Creating Three-way Replicated Volumes” and Section 5.7.2, “Creating Three-way Distributed Replicated Volumes”.

Two-way replicated volume creates two copies of files across the bricks in the volume. The number of bricks must be multiple of two for a replicated volume. To protect against server and disk failures, it is recommended that the bricks of the volume are from different servers.

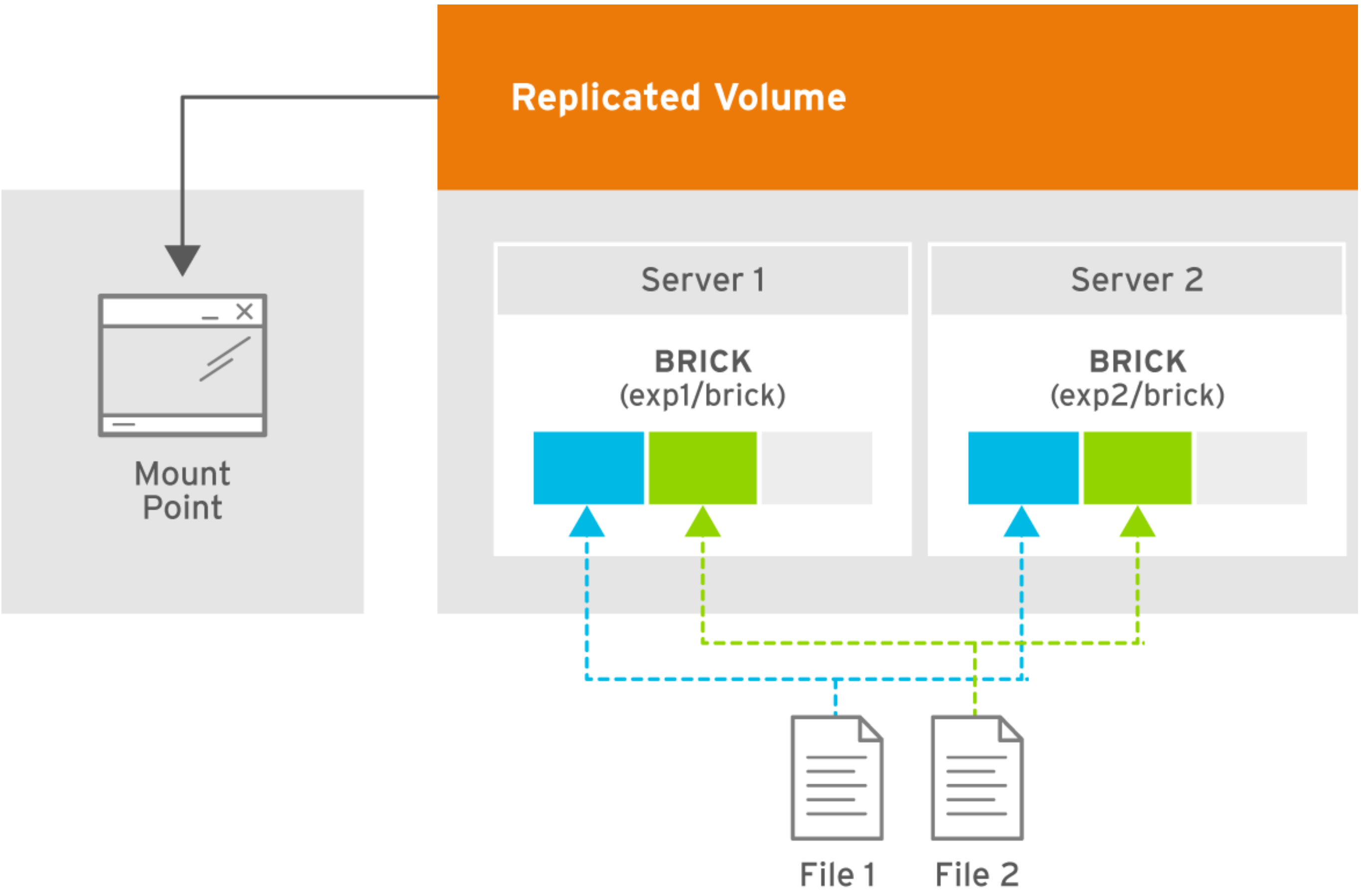


Figure 5.3. Illustration of a Two-way Replicated Volume

Creating two-way replicated volumes

1. Run the **`gluster volume create`** command to create the replicated volume.

   The syntax is **`# gluster volume create`** ***`NEW-VOLNAME`*** **`[replica`** ***`COUNT`*** **`] [transport tcp | rdma | tcp,rdma]`** ***`NEW-BRICK...`***

   The default value for transport is **`tcp`**. Other options can be passed such as **`auth.allow`** or **`auth.reject`**. See Section 11.1, “Configuring Volume Options” for a full list of parameters.

   > **Example 5.3. Replicated Volume with Two Storage Servers**
   >
   > The order in which bricks are specified determines how they are replicated with each other. For example, every **2** bricks, where **2** is the replica count, forms a replica set. This is illustrated in Figure 5.3, “Illustration of a Two-way Replicated Volume” .
   >
   > ```
   > # gluster volume create test-volume replica 2 transport tcp
   > server1:/rhgs/brick1 server2:/rhgs/brick2
   > Creation of test-volume has been successful
   > Please start the volume to access data.
   > ```

2. Run **`# gluster volume start`** ***`VOLNAME`*** to start the volume.

   ```
   # gluster volume start test-volume
   Starting test-volume has been successful
   ```

3. Run **`gluster volume info`** command to optionally display the volume information.

> **Important**
>
> You must set client-side quorum on replicated volumes to prevent split-brain scenarios. For more information on setting client-side quorum, see Section 11.15.1.2, “Configuring Client-Side Quorum”

5.6.2. Creating Three-way Replicated Volumes

Three-way replicated volume creates three copies of files across multiple bricks in the volume. The number of bricks must be equal to the replica count for a replicated volume. To protect against server and disk failures, it is recommended that the bricks of the volume are from different servers.

Synchronous three-way replication is now fully supported in Red Hat Gluster Storage. It is recommended that three-way replicated volumes use JBOD, but use of hardware RAID with three-way replicated volumes is also supported.

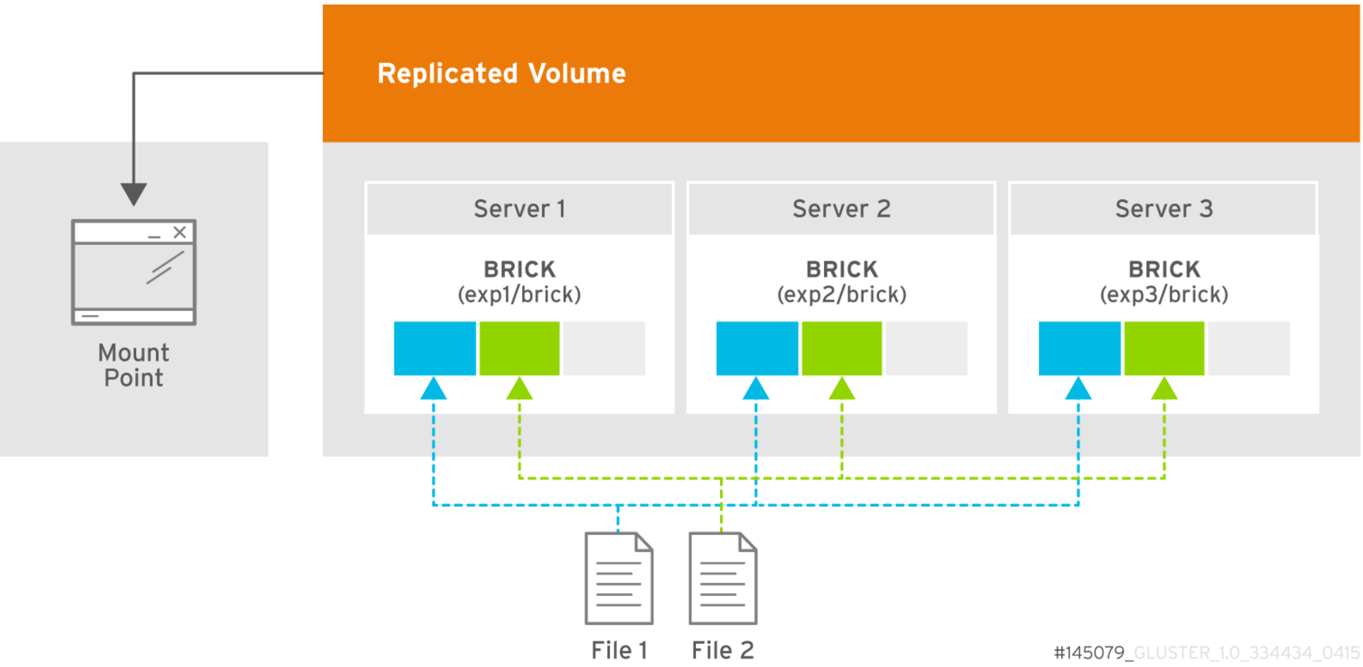


Figure 5.4. Illustration of a Three-way Replicated Volume

Creating three-way replicated volumes

1. Run the **`gluster volume create`** command to create the replicated volume.

   The syntax is **`# gluster volume create`** ***`NEW-VOLNAME`*** **`[replica`** ***`COUNT`*****`] [transport tcp | rdma | tcp,rdma]`** ***`NEW-BRICK...`***

   The default value for transport is **`tcp`**. Other options can be passed such as **`auth.allow`** or **`auth.reject`**. See Section 11.1, “Configuring Volume Options” for a full list of parameters.

   > **Example 5.4. Replicated Volume with Three Storage Servers**

> The order in which bricks are specified determines how bricks are replicated with each other. For example, every **n** bricks, where **3** is the replica count forms a replica set. This is illustrated in Figure 5.4, “Illustration of a Three-way Replicated Volume”.
>
> ```
> # gluster volume create test-volume replica 3 transport tcp
> server1:/rhgs/brick1 server2:/rhgs/brick2 server3:/rhgs/brick3
> Creation of test-volume has been successful
> Please start the volume to access data.
> ```

2. Run `# gluster volume start` ***`VOLNAME`*** to start the volume.

   ```
   # gluster volume start test-volume
   Starting test-volume has been successful
   ```

3. Run **`gluster volume info`** command to optionally display the volume information.

Important

By default, the client-side quorum is enabled on three-way replicated volumes to minimize split-brain scenarios. For more information on client-side quorum, see Section 11.15.1.2, “Configuring Client-Side Quorum”

5.6.3. Creating Sharded Replicated Volumes

Sharding breaks files into smaller pieces so that they can be distributed across the bricks that comprise a volume. This is enabled on a per-volume basis.

When sharding is enabled, files written to a volume are divided into pieces. The size of the pieces depends on the value of the volume's features.shard-block-size parameter. The first piece is written to a brick and given a GFID like a normal file. Subsequent pieces are distributed evenly between bricks in the volume (sharded bricks are distributed by default), but they are written to that brick's **`.shard`** directory, and are named with the GFID and a number indicating the order of the pieces. For example, if a file is split into four pieces, the first piece is named GFID and stored normally. The other three pieces are named GFID.1, GFID.2, and GFID.3 respectively. They are placed in the **`.shard`** directory and distributed evenly between the various bricks in the volume.

Because sharding distributes files across the bricks in a volume, it lets you store files with a larger aggregate size than any individual brick in the volume. Because the file pieces are smaller, heal operations are faster, and geo-replicated deployments can sync the small pieces of a file that have changed, rather than syncing the entire aggregate file.

Sharding also lets you increase volume capacity by adding bricks to a volume in an ad-hoc fashion.

5.6.3.1. Supported use cases

Sharding has one supported use case: in the context of providing Red Hat Gluster Storage as a storage domain for Red Hat Enterprise Virtualization, to provide storage for live virtual machine images. Note that sharding is also a requirement for this use case, as it provides significant performance improvements over previous implementations.

Important

Quotas are not compatible with sharding.

Important

Sharding is supported in new deployments only, as there is currently no upgrade path for this feature.

Example 5.5. Example: Three-way replicated sharded volume

1. Set up a three-way replicated volume, as described in the Red Hat Gluster Storage *Administration Guide*: https://access.redhat.com/documentation/en-US/red_hat_gluster_storage/3.4/html/Administration_Guide/sect-Creating_Replicated_Volumes.html#Creating_Three-way_Replicated_Volumes.

2. Before you start your volume, enable sharding on the volume.

```
# gluster volume set test-volume features.shard enable
```

3. Start the volume and ensure it is working as expected.

```
# gluster volume test-volume start
# gluster volume info test-volume
```

5.6.3.2. Configuration Options

Sharding is enabled and configured at the volume level. The configuration options are as follows.

`features.shard`

Enables or disables sharding on a specified volume. Valid values are **`enable`** and **`disable`**. The default value is **`disable`**.

```
# gluster volume set volname features.shard enable
```

Note that this only affects files created after this command is run; files created before this command is run retain their old behaviour.

`features.shard-block-size`

Specifies the maximum size of the file pieces when sharding is enabled. The supported value for this parameter is 512MB.

```
# gluster volume set volname features.shard-block-size 32MB
```

Note that this only affects files created after this command is run; files created before this command is run retain their old behaviour.

5.6.3.3. Finding the pieces of a sharded file

When you enable sharding, you might want to check that it is working correctly, or see how a particular file has been sharded across your volume.

To find the pieces of a file, you need to know that file's GFID. To obtain a file's GFID, run:

```
# getfattr -d -m. -e hex path_to_file
```

Once you have the GFID, you can run the following command on your bricks to see how this file has been distributed:

```
# ls /rhgs/*/.shard -lh | grep GFID
```

5.7. Creating Distributed Replicated Volumes

Use distributed replicated volumes in environments where the requirement to scale storage, and high-reliability is critical. Distributed replicated volumes also offer improved read performance in most environments.

Note

The number of bricks must be a multiple of the replica count for a distributed replicated volume. Also, the order in which bricks are specified has a great effect on data protection. Each replica_count consecutive bricks in the list you give will form a replica set, with all replica sets combined into a distribute set. To ensure that replica-set members are not placed on the same node, list the first brick on every server, then the second brick on every server in the same order, and so on.

Prerequisites

- A trusted storage pool has been created, as described in Section 4.1, “Adding Servers to the Trusted Storage Pool”.
- Understand how to start and stop volumes, as described in Section 5.11, “Starting Volumes”.

5.7.1. Creating Two-way Distributed Replicated Volumes

Warning

Support for two-way replication is planned for deprecation and removal in future versions of Red Hat Gluster Storage. This will affect both replicated and distributed-replicated volumes.

Support is being removed because two-way replication does not provide adequate protection from split-brain conditions. While a dummy node can be used as an interim solution for this problem, Red Hat recommends that all volumes that currently use two-way replication are migrated to use either arbitrated replication or three-way replication.

Instructions for migrating a two-way replicated volume to an arbitrated replicated volume are available in Section 5.8.5, “Converting to an arbitrated volume”.

Information about three-way replication is available in Section 5.6.2, “Creating Three-way Replicated Volumes” and Section 5.7.2, “Creating Three-way Distributed Replicated Volumes”.

Two-way distributed replicated volumes distribute and create two copies of files across the bricks in a volume. The number of bricks must be multiple of the replica count for a replicated volume. To protect against server and disk failures, the bricks of the volume should be from different servers.

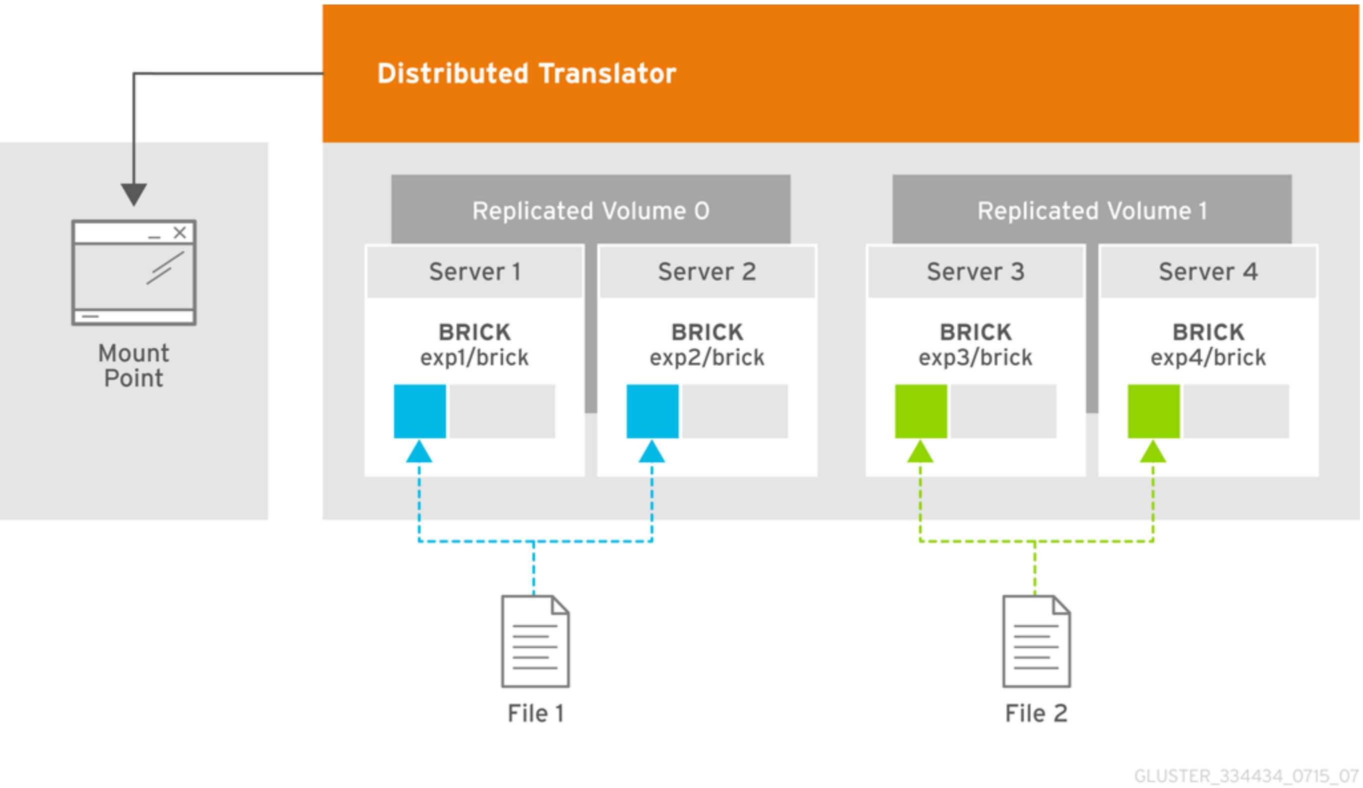


Figure 5.5. Illustration of a Two-way Distributed Replicated Volume

Creating two-way distributed replicated volumes

1. Run the **`gluster volume create`** command to create the distributed replicated volume.

   The syntax is **`# gluster volume create`** ***`NEW-VOLNAME`*** **`[replica`** ***`COUNT`*** **`] [transport tcp | rdma | tcp,rdma]`** ***`NEW-BRICK...`***

   The default value for transport is **`tcp`**. Other options can be passed such as **`auth.allow`** or **`auth.reject`**. See Section 11.1, “Configuring Volume Options” for a full list of parameters.

Example 5.6. Four Node Distributed Replicated Volume with a Two-way Replication

The order in which bricks are specified determines how they are replicated with each other. For example, the first two bricks specified replicate each other where *2* is the replica *count*.

```
# gluster volume create test-volume replica 2 transport tcp
server1:/rhgs/brick1 server2:/rhgs/brick1 server3:/rhgs/brick1
server4:/rhgs/brick1
Creation of test-volume has been successful
Please start the volume to access data.
```

Example 5.7. Six Node Distributed Replicated Volume with a Two-way Replication

```
# gluster volume create test-volume replica 2 transport tcp
server1:/rhgs/brick1 server2:/rhgs/brick1 server3:/rhgs/brick1
server4:/rhgs/brick1 server5:/rhgs/brick1 server6:/rhgs/brick1
Creation of test-volume has been successful
Please start the volume to access data.
```

2. Run `# gluster volume start` ***`VOLNAME`*** to start the volume.

```
# gluster volume start test-volume
Starting test-volume has been successful
```

3. Run `gluster volume info` command to optionally display the volume information.

Important

You must ensure to set server-side quorum and client-side quorum on the distributed-replicated volumes to prevent split-brain scenarios. For more information on setting quorums, see Section 11.15.1, “Preventing Split-brain”

5.7.2. Creating Three-way Distributed Replicated Volumes

Three-way distributed replicated volume distributes and creates three copies of files across multiple bricks in the volume. The number of bricks must be equal to the replica count for a replicated volume. To protect against server and disk failures, it is recommended that the bricks of the volume are from different servers.

Synchronous three-way distributed replication is now fully supported in Red Hat Gluster Storage. It is recommended that three-way distributed replicated volumes use JBOD, but use of hardware RAID with three-way distributed replicated volumes is also supported.

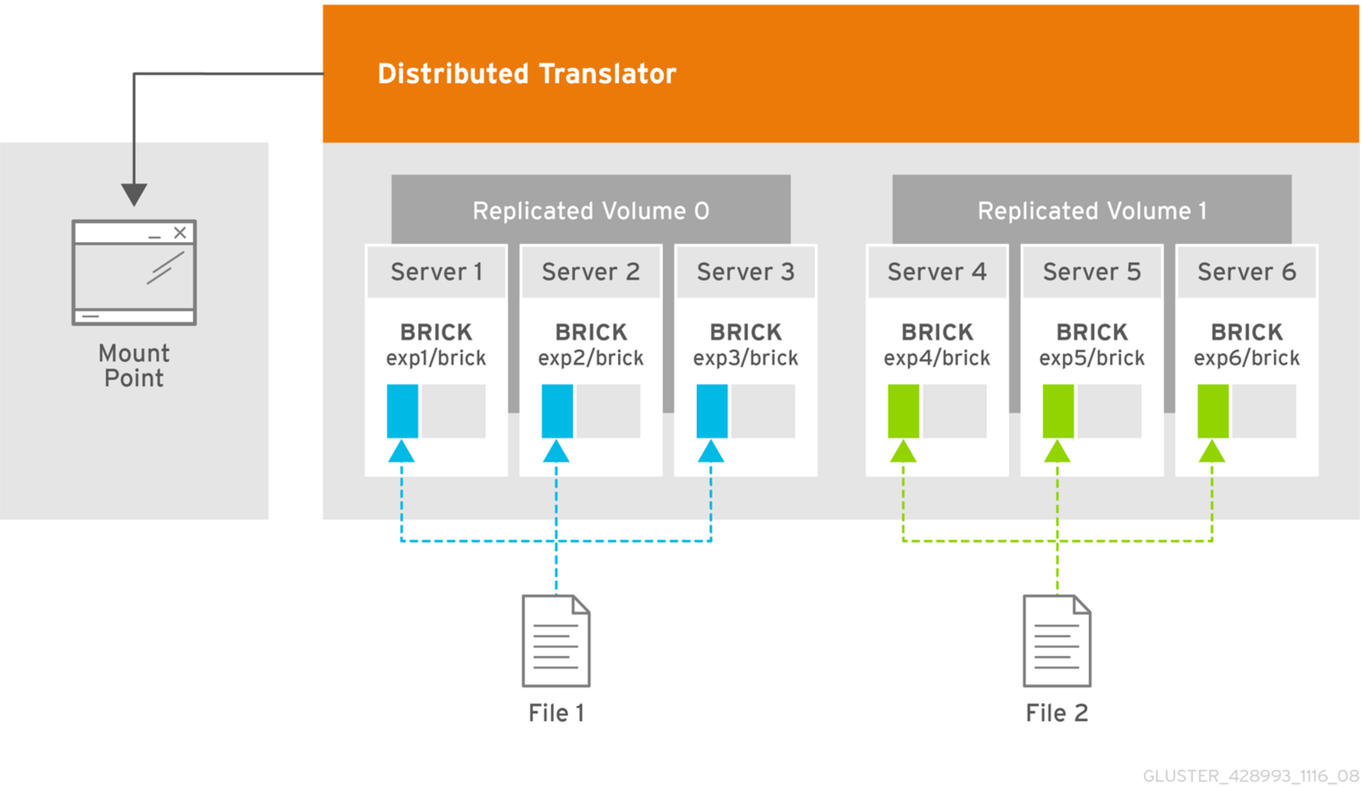


Figure 5.6. Illustration of a Three-way Distributed Replicated Volume

Creating three-way distributed replicated volumes

1. Run the **`gluster volume create`** command to create the distributed replicated volume.

   The syntax is **`# gluster volume create`** ***`NEW-VOLNAME`*** **`[replica`** ***`COUNT`*** **`] [transport tcp | rdma | tcp,rdma]`** ***`NEW-BRICK...`***

   The default value for transport is **`tcp`**. Other options can be passed such as **`auth.allow`** or **`auth.reject`**. See Section 11.1, “Configuring Volume Options” for a full list of parameters.

   > **Example 5.8. Six Node Distributed Replicated Volume with a Three-way Replication**
   >
   > The order in which bricks are specified determines how bricks are replicated with each other. For example, first *3* bricks, where *3* is the replica *count* forms a replicate set.
   >
   > ```
   > # gluster volume create test-volume replica 3 transport tcp
   > server1:/rhgs/brick1 server2:/rhgs/brick1 server3:/rhgs/brick1
   > server4:/rhgs/brick1 server5:/rhgs/brick1 server6:/rhgs/brick1
   > Creation of test-volume has been successful
   > Please start the volume to access data.
   > ```

2. Run **`# gluster volume start`** ***`VOLNAME`*** to start the volume.

   ```
   # gluster volume start test-volume
   Starting test-volume has been successful
   ```

3. Run **`gluster volume info`** command to optionally display the volume information.

Important

By default, the client-side quorum is enabled on three-way distributed replicated volumes. You must also set server-side quorum on the distributed-replicated volumes to prevent split-brain scenarios. For more information on setting quorums, see Section 11.15.1, “Preventing Split-brain”.

5.8. Creating Arbitrated Replicated Volumes

An arbitrated replicated volume is similar to a two-way replicated volume, in that it contains two full copies of the files in the volume. Arbitrated volumes have an extra *arbiter* brick for every two data bricks in the volume. Arbiter bricks do not store file data; they only store file names, structure, and metadata. Arbiter bricks use client quorum to compare metadata on the arbiter with the metadata of the other nodes to ensure consistency in the volume and prevent split-brain conditions.

Advantages of arbitrated replicated volumes

Better consistency

When an arbiter is configured, arbitration logic uses client-side quorum in auto mode to prevent file operations that would lead to split-brain conditions.

Less disk space required

Because an arbiter brick only stores file names and metadata, an arbiter brick can be much smaller than the other bricks in the volume.

Fewer nodes required

The node that contains the arbiter brick of one volume can be configured with the data brick of another volume. This "chaining" configuration allows you to use fewer nodes to fulfill your overall storage requirements.

Easy migration from deprecated two-way replicated volumes

Red Hat Gluster Storage can convert a two-way replicated volume without arbiter bricks into an arbitrated replicated volume. See Section 5.8.5, “Converting to an arbitrated volume” for details.

Limitations of arbitrated replicated volumes

- Arbitrated replicated volumes provide better data consistency than a two-way replicated volume that does not have arbiter bricks. However, because arbitrated replicated volumes store only metadata, they provide the same level of availability as a two-way replicated volume that does not have arbiter bricks. To achieve high-availability, you need to use a three-way replicated volume instead of an arbitrated replicated volume.
- Tiering is not compatible with arbitrated replicated volumes.
- Arbitrated volumes can only be configured in sets of three bricks at a time. Red Hat Gluster Storage can convert an existing two-way replicated volume without arbiter bricks into an arbitrated replicated volume by adding an arbiter brick to that volume. See Section 5.8.5, “Converting to an arbitrated volume” for details.

5.8.1. Arbitrated volume requirements

This section outlines the requirements of a supported arbitrated volume deployment.

5.8.1.1. System requirements for nodes hosting arbiter bricks

The minimum system requirements for a node that contains an arbiter brick differ depending on the configuration choices made by the administrator. See Section 5.8.4, “Creating multiple arbitrated replicated volumes across fewer total nodes” for details about the differences between the dedicated arbiter and chained arbiter configurations.

Table 5.1. Requirements for arbitrated configurations on physical machines

Configuration type	Min CPU	Min RAM	NIC	Arbiter Brick Size	Max Latency
Dedicated arbiter	64-bit quad-core processor with 2 sockets	8 GB [a]	Match to other nodes in the storage pool	1 TB to 4 TB [b]	5 ms
Chained arbiter	Match to other nodes in the storage pool			1 TB to 4 TB [c]	5 ms

[a] More RAM may be necessary depending on the combined capacity of the number of arbiter bricks on the node.

[b] Arbiter and data bricks can be configured on the same device provided that the data and arbiter bricks belong to different replica sets. See Section 5.8.1.2, “Arbiter capacity requirements” for further details on sizing arbiter volumes.

[c] Multiple bricks can be created on a single RAIDed physical device. Please refer the following product documentation: Section 21.2, “Brick Configuration”

The requirements for arbitrated configurations on virtual machines are:

- minimum 4 vCPUs
- minimum 16 GB RAM
- 1 TB to 4 TB of virtual disk space
- maximum 5 ms latency

5.8.1.2. Arbiter capacity requirements

Because an arbiter brick only stores file names and metadata, an arbiter brick can be much smaller than the other bricks in the volume or replica set. The required size for an arbiter brick depends on the number of files being stored on the volume.

The recommended minimum arbiter brick size can be calculated with the following formula:

```
minimum arbiter brick size = 4 KB * ( size in KB of largest data brick in
volume or replica set / average file size in KB)
```

For example, if you have two 1 TB data bricks, and the average size of the files is 2 GB, then the recommended minimum size for your arbiter brick 2 MB, as shown in the following example:

```
minimum arbiter brick size  = 4 KB * ( 1 TB / 2 GB )
                            = 4 KB * ( 1000000000 KB / 2000000 KB )
                            = 4 KB * 500 KB
                            = 2000 KB
                            = 2 MB
```

If sharding is enabled, and your shard-block-size is smaller than the average file size in KB, then you need to use the following formula instead, because each shard also has a metadata file:

```
minimum arbiter brick size = 4 KB * ( size in KB of largest data brick in
volume or replica set / shard block size in KB )
```

Alternatively, if you know how many files you will store in a volume, the recommended minimum arbiter brick size is the maximum number of files multiplied by 4 KB. For example, if you expect to have 200,000 files on your volume, your arbiter brick should be at least 800,000 KB, or 0.8 GB, in size.

Red Hat also recommends overprovisioning where possible so that there is no short-term need to increase the size of the arbiter brick.

5.8.2. Arbitration logic

In an arbitrated volume, whether a file operation is permitted depends on the current state of the bricks in the volume. The following table describes arbitration behavior in all possible volume states.

Table 5.2. Allowed operations for current volume state

Volume state	Arbitration behavior
All bricks available	All file operations permitted.
Arbiter and 1 data brick available	If the arbiter does not agree with the available data node, write operations fail with ENOTCONN (since the brick that is correct is not available). Other file operations are permitted. If the arbiter's metadata agrees with the available data node, all file operations are permitted.
Arbiter down, data bricks available	All file operations are permitted. The arbiter's records are healed when it becomes available.
Only one brick available	If the available brick is a data brick, client quorum is not met, and the volume enters an EROFS state. If the available brick is the arbiter, all file operations fail with ENOTCONN.

5.8.3. Creating an arbitrated replicated volume

The command for creating an arbitrated replicated volume has the following syntax:

```
# gluster volume create VOLNAME replica 3 arbiter 1 HOST1:DATA_BRICK1
HOST2:DATA_BRICK2 HOST3:ARBITER_BRICK3
```

This creates a volume with one arbiter for every three replicate bricks. The arbiter is the last brick in every set of three bricks.

Note

The syntax of this command is misleading. There are a total of 3 bricks in this set. This command creates a volume with two bricks that replicate all data and one arbiter brick that replicates only metadata.

In the following example, the bricks on server3 and server6 are the arbiter bricks. Note that because multiple sets of three bricks are provided, this creates a distributed replicated volume with arbiter bricks.

```
# gluster volume create testvol replica 3 arbiter 1 \
server1:/bricks/brick server2:/bricks/brick server3:/bricks/arbiter_brick \
server4:/bricks/brick server5:/bricks/brick server6:/bricks/arbiter_brick
```

```
# gluster volume info testvol
Volume Name: testvol
Type: Distributed-Replicate
Volume ID: ed9fa4d5-37f1-49bb-83c3-925e90fab1bc
Status: Created
Snapshot Count: 0
Number of Bricks: 2 x (2 + 1) = 6
Transport-type: tcp
Bricks:
Brick1: server1:/bricks/brick
Brick2: server2:/bricks/brick
Brick3: server3:/bricks/arbiter_brick (arbiter)
Brick1: server4:/bricks/brick
Brick2: server5:/bricks/brick
Brick3: server6:/bricks/arbiter_brick (arbiter)
Options Reconfigured:
transport.address-family: inet
performance.readdir-ahead: on
nfs.disable: on
```

5.8.4. Creating multiple arbitrated replicated volumes across fewer total nodes

If you are configuring more than one arbitrated-replicated volume, or a single volume with multiple replica sets, you can use fewer nodes in total by using either of the following techniques:

- Chain multiple arbitrated replicated volumes together, by placing the arbiter brick for one volume on the same node as a data brick for another volume. Chaining is useful for write-heavy workloads when file size is closer to metadata file size (that is, from 32–128 KiB). This avoids all metadata I/O going through a single disk.

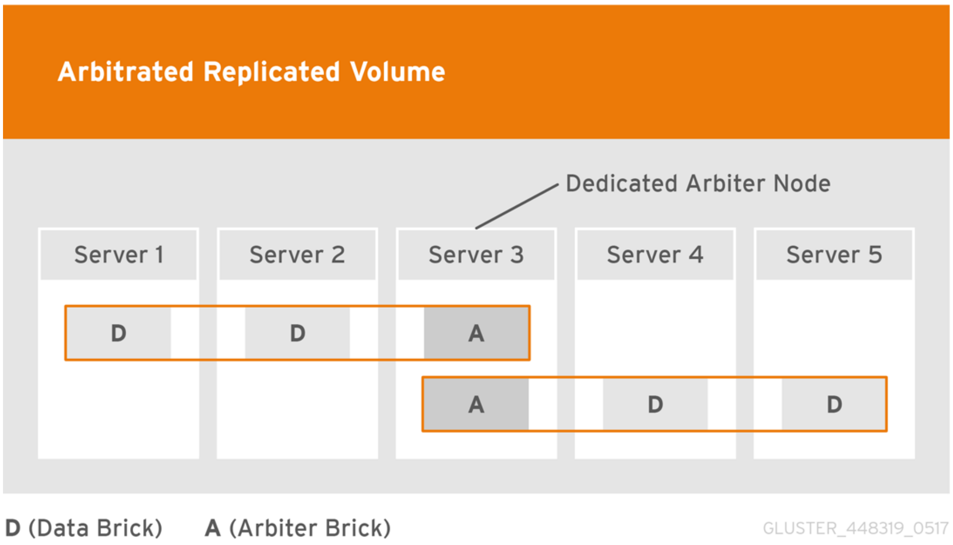
  In arbitrated distributed-replicated volumes, you can also place an arbiter brick on the same node as another replica sub-volume's data brick, since these do not share the same data.

- Place the arbiter bricks from multiple volumes on a single dedicated node. A dedicated arbiter node is suited to write-heavy workloads with larger files, and read-heavy workloads.

Example 5.9. Example of a dedicated configuration

The following commands create two arbitrated replicated volumes, firstvol and secondvol. Server3 contains the arbiter bricks of both volumes.

```
# gluster volume create firstvol replica 3 arbiter 1 server1:/bricks/brick
server2:/bricks/brick server3:/bricks/arbiter_brick
# gluster volume create secondvol replica 3 arbiter 1
server4:/bricks/data_brick server5:/bricks/brick server3:/bricks/brick
```

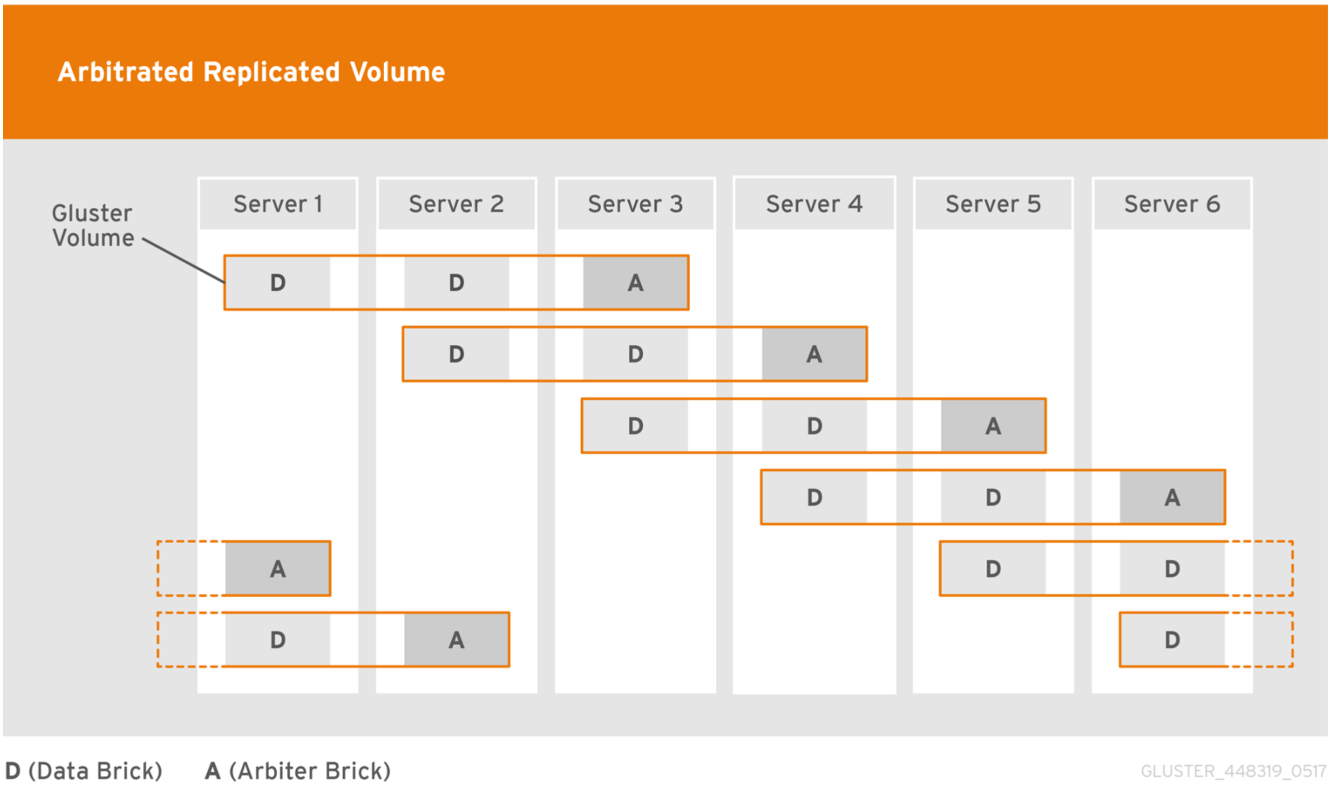


Two gluster volumes configured across five servers to create two three-way arbitrated replicated volumes, with the arbiter bricks on a dedicated arbiter node.

Example 5.10. Example of a chained configuration

The following command configures an arbitrated replicated volume with six sub-volumes chained across six servers in a 6 x (2 + 1) configuration.

```
# gluster volume create arbrepvol replica 3 arbiter 1
server1:/bricks/brick1 server2:/bricks/brick1
server3:/bricks/arbiter_brick1 server2:/bricks/brick2
server3:/bricks/brick2 server4:/bricks/arbiter_brick2
server3:/bricks/brick3 server4:/bricks/brick3
server5:/bricks/arbiter_brick3 server4:/bricks/brick4
server5:/bricks/brick4 server6:/bricks/arbiter_brick4
server5:/bricks/brick5 server6:/bricks/brick5
server1:/bricks/arbiter_brick5 server6:/bricks/brick6
server1:/bricks/brick6 server2:/bricks/arbiter_brick6
```



Six replicated gluster sub-volumes chained across six servers to create a 6 * (2 + 1) arbitrated distributed-replicated configuration.

5.8.5. Converting to an arbitrated volume

You can convert a replicated volume into an arbitrated volume by adding new arbiter bricks for each replicated sub-volume, or replacing replica bricks with arbiter bricks.

- Procedure 5.1, “Converting a replica 2 volume to an arbitrated volume”
- Procedure 5.2, “Converting a replica 3 volume to an arbitrated volume”

Procedure 5.1. Converting a replica 2 volume to an arbitrated volume

1. **Verify that healing is not in progress**

   ```
   # gluster volume heal VOLNAME info
   ```

   Wait until pending heal entries is `0` before proceeding.

2. **Disable and stop self-healing**

   Run the following commands to disable data, metadata, and entry self-heal, and the self-heal daemon.

   ```
   # gluster volume set VOLNAME cluster.data-self-heal off
   # gluster volume set VOLNAME cluster.metadata-self-heal off
   # gluster volume set VOLNAME cluster.entry-self-heal off
   # gluster volume set VOLNAME self-heal-daemon off
   ```

3. **Add arbiter bricks to the volume**

Convert the volume by adding an arbiter brick for each replicated sub-volume.

```
# gluster volume add-brick VOLNAME replica 3 arbiter 1 HOST:arbiter-
brick-path
```

For example, if you have an existing two-way replicated volume called testvol, and a new brick for the arbiter to use, you can add a brick as an arbiter with the following command:

```
# gluster volume add-brick testvol replica 3 arbiter 1
server:/bricks/arbiter_brick
```

If you have an existing two-way distributed-replicated volume, you need a new brick for each sub-volume in order to convert it to an arbitrated distributed-replicated volume, for example:

```
# gluster volume add-brick testvol replica 3 arbiter 1
server1:/bricks/arbiter_brick1 server2:/bricks/arbiter_brick2
```

4. **Wait for client volfiles to update**

   This takes about 5 minutes.

5. **Verify that bricks added successfully**

   ```
   # gluster volume info VOLNAME
   # gluster volume status VOLNAME
   ```

6. **Re-enable self-healing**

   Run the following commands to re-enable self-healing on the servers.

   ```
   # gluster volume set VOLNAME cluster.data-self-heal on
   # gluster volume set VOLNAME cluster.metadata-self-heal on
   # gluster volume set VOLNAME cluster.entry-self-heal on
   # gluster volume set VOLNAME self-heal-daemon on
   ```

7. **Verify all entries are healed**

   ```
   # gluster volume heal VOLNAME info
   ```

   Wait until pending heal entries is **0** to ensure that all heals completed successfully.

Procedure 5.2. Converting a replica 3 volume to an arbitrated volume

1. **Verify that healing is not in progress**

   ```
   # gluster volume heal VOLNAME info
   ```

   Wait until pending heal entries is **0** before proceeding.

2. **Reduce the replica count of the volume to 2**

Remove one brick from every sub-volume in the volume so that the replica count is reduced to 2. For example, in a replica 3 volume that distributes data across 2 sub-volumes, run the following command:

```
# gluster volume remove-brick VOLNAME replica 2 HOST:subvol1-brick-
path HOST:subvol2-brick-path force
```

Note

In a distributed replicated volume, data is distributed across sub-volumes, and replicated across bricks in a sub-volume. This means that to reduce the replica count of a volume, you need to remove a brick from every sub-volume.

Bricks are grouped by sub-volume in the **`gluster volume info`** output. If the replica count is 3, the first 3 bricks form the first sub-volume, the next 3 bricks form the second sub-volume, and so on.

```
# gluster volume info VOLNAME
[...]
Number of Bricks: 2 x 3 = 6
Transport-type: tcp
Bricks:
Brick1: node1:/test1/brick
Brick2: node2:/test2/brick
Brick3: node3:/test3/brick
Brick4: node1:/test4/brick
Brick5: node2:/test5/brick
Brick6: node3:/test6/brick
[...]
```

In this volume, data is distributed across two sub-volumes, which each consist of three bricks. The first sub-volume consists of bricks 1, 2, and 3. The second sub-volume consists of bricks 4, 5, and 6. Removing any one brick from each subvolume using the following command reduces the replica count to 2 as required.

```
# gluster volume remove-brick VOLNAME replica 2 HOST:subvol1-
brick-path HOST:subvol2-brick-path force
```

3. **Disable and stop self-healing**

   Run the following commands to disable data, metadata, and entry self-heal, and the self-heal daemon.

   ```
   # gluster volume set VOLNAME cluster.data-self-heal off
   # gluster volume set VOLNAME cluster.metadata-self-heal off
   # gluster volume set VOLNAME cluster.entry-self-heal off
   # gluster volume set VOLNAME self-heal-daemon off
   ```

4. **Add arbiter bricks to the volume**

   Convert the volume by adding an arbiter brick for each replicated sub-volume.

```
# gluster volume add-brick VOLNAME replica 3 arbiter 1 HOST:arbiter-
brick-path
```

For example, if you have an existing replicated volume:

```
# gluster volume add-brick testvol replica 3 arbiter 1
server:/bricks/brick
```

If you have an existing distributed-replicated volume:

```
# gluster volume add-brick testvol replica 3 arbiter 1
server1:/bricks/arbiter_brick1 server2:/bricks/arbiter_brick2
```

5. **Wait for client volfiles to update**

   This takes about 5 minutes. Verify that this is complete by running the following command on each client.

   ```
   # grep -ir connected mount-path/.meta/graphs/active/volname-client-
   */private
   ```

   The number of times **`connected=1`** appears in the output is the number of bricks connected to the client.

6. **Verify that bricks added successfully**

   ```
   # gluster volume info VOLNAME
   # gluster volume status VOLNAME
   ```

7. **Re-enable self-healing**

   Run the following commands to re-enable self-healing on the servers.

   ```
   # gluster volume set VOLNAME cluster.data-self-heal on
   # gluster volume set VOLNAME cluster.metadata-self-heal on
   # gluster volume set VOLNAME cluster.entry-self-heal on
   # gluster volume set VOLNAME self-heal-daemon on
   ```

8. **Verify all entries are healed**

   ```
   # gluster volume heal VOLNAME info
   ```

   Wait until pending heal entries is **`0`** to ensure that all heals completed successfully.

5.8.6. Converting an arbitrated volume to a three-way replicated volume

You can convert an arbitrated volume into a three-way replicated volume or a three-way distributed replicated volume by replacing the arbiter bricks with full bricks for each replicated sub-volume.

Procedure 5.3. Converting an arbitrated volume to a replica 3 volume

1. **Verify that healing is not in progress**

```
# gluster volume heal VOLNAME info
```

Wait until pending heal entries is `0` before proceeding.

2. **Remove arbiter bricks from the volume**

   Check which bricks are listed as `(arbiter)`, and then remove those bricks from the volume.

   ```
   # gluster volume info VOLNAME
   ```

   ```
   # gluster volume remove-brick VOLNAME replica 2 HOST:arbiter-brick-path force
   ```

3. **Disable and stop self-healing**

   Run the following commands to disable data, metadata, and entry self-heal, and the self-heal daemon.

   ```
   # gluster volume set VOLNAME cluster.data-self-heal off
   # gluster volume set VOLNAME cluster.metadata-self-heal off
   # gluster volume set VOLNAME cluster.entry-self-heal off
   # gluster volume set VOLNAME self-heal-daemon off
   ```

4. **Add full bricks to the volume**

   Convert the volume by adding a brick for each replicated sub-volume.

   ```
   # gluster volume add-brick VOLNAME replica 3 HOST:brick-path
   ```

   For example, if you have an existing arbitrated replicated volume:

   ```
   # gluster volume add-brick testvol replica 3 server:/bricks/brick
   ```

   If you have an existing arbitrated distributed-replicated volume:

   ```
   # gluster volume add-brick testvol replica 3 server1:/bricks/brick1
   server2:/bricks/brick2
   ```

5. **Wait for client volfiles to update**

   This takes about 5 minutes.

6. **Verify that bricks added successfully**

   ```
   # gluster volume info VOLNAME
   # gluster volume status VOLNAME
   ```

7. **(Optional) Re-enable self-healing**

   Run the following commands to re-enable self-healing on the servers.

   ```
   # gluster volume set VOLNAME cluster.data-self-heal on
   ```

```
# gluster volume set VOLNAME cluster.metadata-self-heal on
# gluster volume set VOLNAME cluster.entry-self-heal on
# gluster volume set VOLNAME self-heal-daemon on
```

8. **Verify all entries are healed**

   ```
   # gluster volume heal VOLNAME info
   ```

   Wait until pending heal entries is `0` to ensure that all heals completed successfully.

5.8.7. Tuning recommendations for arbitrated volumes

Red Hat recommends the following when arbitrated volumes are in use:

- For dedicated arbiter nodes, use JBOD for arbiter bricks, and RAID6 for data bricks.
- For chained arbiter volumes, use the same RAID6 drive for both data and arbiter bricks.

See Chapter 21, *Tuning for Performance* for more information on enhancing performance that is not specific to the use of arbiter volumes.

5.9. Creating Dispersed Volumes

Dispersed volumes are based on erasure coding. Erasure coding (EC) is a method of data protection in which data is broken into fragments, expanded and encoded with redundant data pieces and stored across a set of different locations. This allows the recovery of the data stored on one or more bricks in case of failure. The number of bricks that can fail without losing data is configured by setting the redundancy count.

Dispersed volume requires less storage space when compared to a replicated volume. It is equivalent to a replicated pool of size two, but requires 1.5 TB instead of 2 TB to store 1 TB of data when the redundancy level is set to 2. In a dispersed volume, each brick stores some portions of data and parity or redundancy. The dispersed volume sustains the loss of data based on the redundancy level.

Important

Dispersed volume configuration is supported only on JBOD storage. For more information, see Section 21.1.2, “JBOD”.

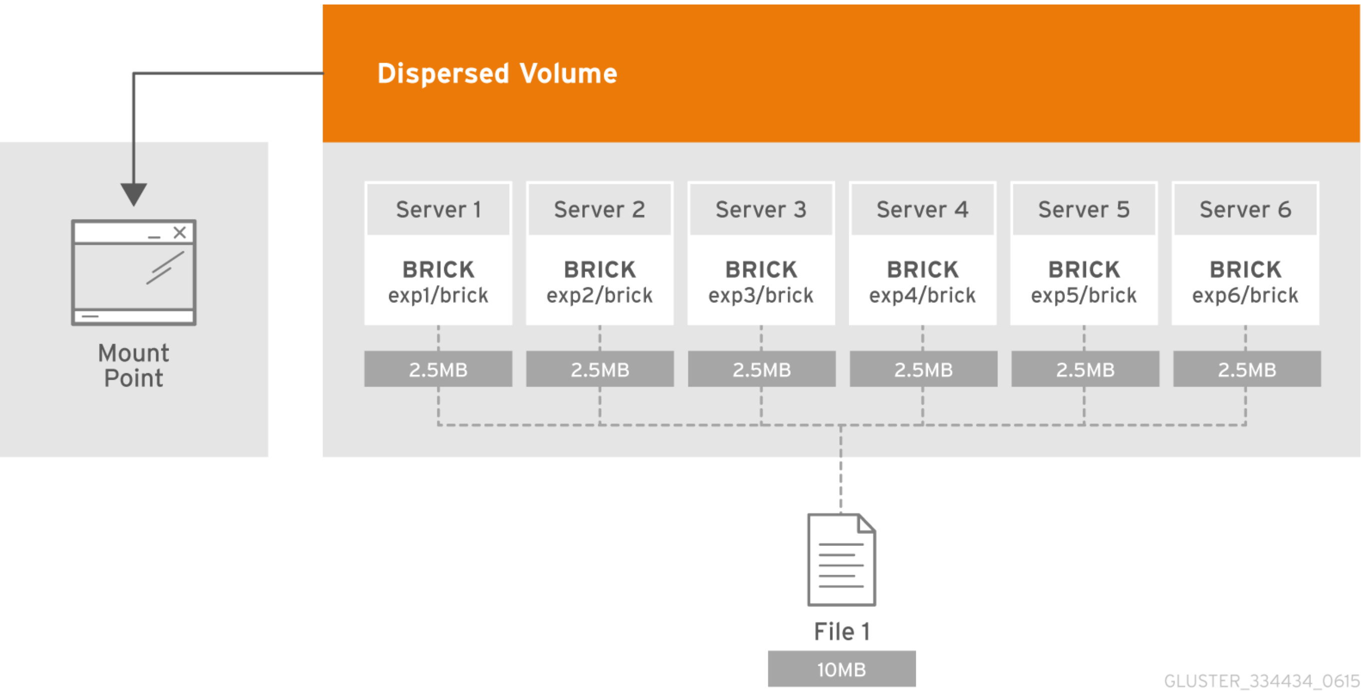


Figure 5.7. Illustration of a Dispersed Volume

The data protection offered by erasure coding can be represented in simple form by the following equation: **`n = k + m`**. Here **`n`** is the total number of bricks, we would require any **`k`** bricks out of **`n`** bricks for recovery. In other words, we can tolerate failure up to any **`m`** bricks. With this release, the following configurations are supported:

- 6 bricks with redundancy level 2 (4 + 2)
- 10 bricks with redundancy level 2 (8 + 2)
- 11 bricks with redundancy level 3 (8 + 3)
- 12 bricks with redundancy level 4 (8 + 4)
- 20 bricks with redundancy level 4 (16 + 4)

For optimal fault tolerance, create each brick on a separate server. Creating multiple bricks on a single server is supported, but the more bricks there are on a single server, the greater the risk to availability and consistency when that single server becomes unavailable.

Use **`gluster volume create`** to create different types of volumes, and **`gluster volume info`** to verify successful volume creation.

Prerequisites

- Create a trusted storage pool as described in Section 4.1, “Adding Servers to the Trusted Storage Pool”.
- Understand how to start and stop volumes, as described in Section 5.11, “Starting Volumes”.

Important

Red Hat recommends you to review the Dispersed Volume configuration recommendations explained in Section 5.9, “Creating Dispersed Volumes” before creating the Dispersed volume.

To Create a dispersed volume

1. Run the **gluster volume create** command to create the dispersed volume.

   The syntax is **# gluster volume create *NEW-VOLNAME* [disperse-data *COUNT*] [redundancy *COUNT*] [transport tcp | rdma | tcp,rdma] *NEW-BRICK...***

   The number of bricks required to create a disperse volume is the sum of **disperse-data *count*** and **redundancy *count***.

   The **disperse-data count** option specifies the number of bricks that is part of the dispersed volume, excluding the count of the redundant bricks. For example, if the total number of bricks is 6 and **redundancy-count** is specified as 2, then the disperse-data count is 4 (6 - 2 = 4). If the **disperse-data count** option is not specified, and only the **redundancy count** option is specified, then the **disperse-data count** is computed automatically by deducting the redundancy count from the specified total number of bricks.

   Redundancy determines how many bricks can be lost without interrupting the operation of the volume. If **redundancy count** is not specified, based on the configuration it is computed automatically to the optimal value and a warning message is displayed.

   The default value for transport is **tcp**. Other options can be passed such as **auth.allow** or **auth.reject**. See Section 5.3, “About Encrypted Disk” for a full list of parameters.

   Example 5.11. Dispersed Volume with Six Storage Servers

   ```
   # gluster volume create test-volume disperse-data 4 redundancy 2
   transport tcp server1:/rhgs1/brick1 server2:/rhgs2/brick2
   server3:/rhgs3/brick3 server4:/rhgs4/brick4 server5:/rhgs5/brick5
   server6:/rhgs6/brick6
   Creation of test-volume has been successful
   Please start the volume to access data.
   ```

2. Run **# gluster volume start *VOLNAME*** to start the volume.

   ```
   # gluster volume start test-volume
   Starting test-volume has been successful
   ```

   Important

   The **open-behind** volume option is enabled by default. If you are accessing the dispersed volume using the SMB protocol, you must disable the **open-behind** volume option to avoid performance bottleneck on large file workload. Run the following command to disable **open-behind** volume option:

   ```
   # gluster volume set VOLNAME open-behind off
   ```

   For information on **open-behind** volume option, see Section 11.1, “Configuring Volume Options”

3. Run **gluster volume info** command to optionally display the volume information.

5.10. Creating Distributed Dispersed Volumes

Distributed dispersed volumes support the same configurations of erasure coding as dispersed volumes. The number of bricks in a distributed dispersed volume must be a multiple of (K+M). With this release, the following configurations are supported:

- Multiple disperse sets containing 6 bricks with redundancy level 2
- Multiple disperse sets containing 10 bricks with redundancy level 2
- Multiple disperse sets containing 11 bricks with redundancy level 3
- Multiple disperse sets containing 12 bricks with redundancy level 4
- Multiple disperse sets containing 20 bricks with redundancy level 4

Important

Distributed dispersed volume configuration is supported only on JBOD storage. For more information, see Section 21.1.2, “JBOD”.

Use **`gluster volume create`** to create different types of volumes, and **`gluster volume info`** to verify successful volume creation.

Prerequisites

- A trusted storage pool has been created, as described in Section 4.1, “Adding Servers to the Trusted Storage Pool”.
- Understand how to start and stop volumes, as described in Section 5.11, “Starting Volumes”.

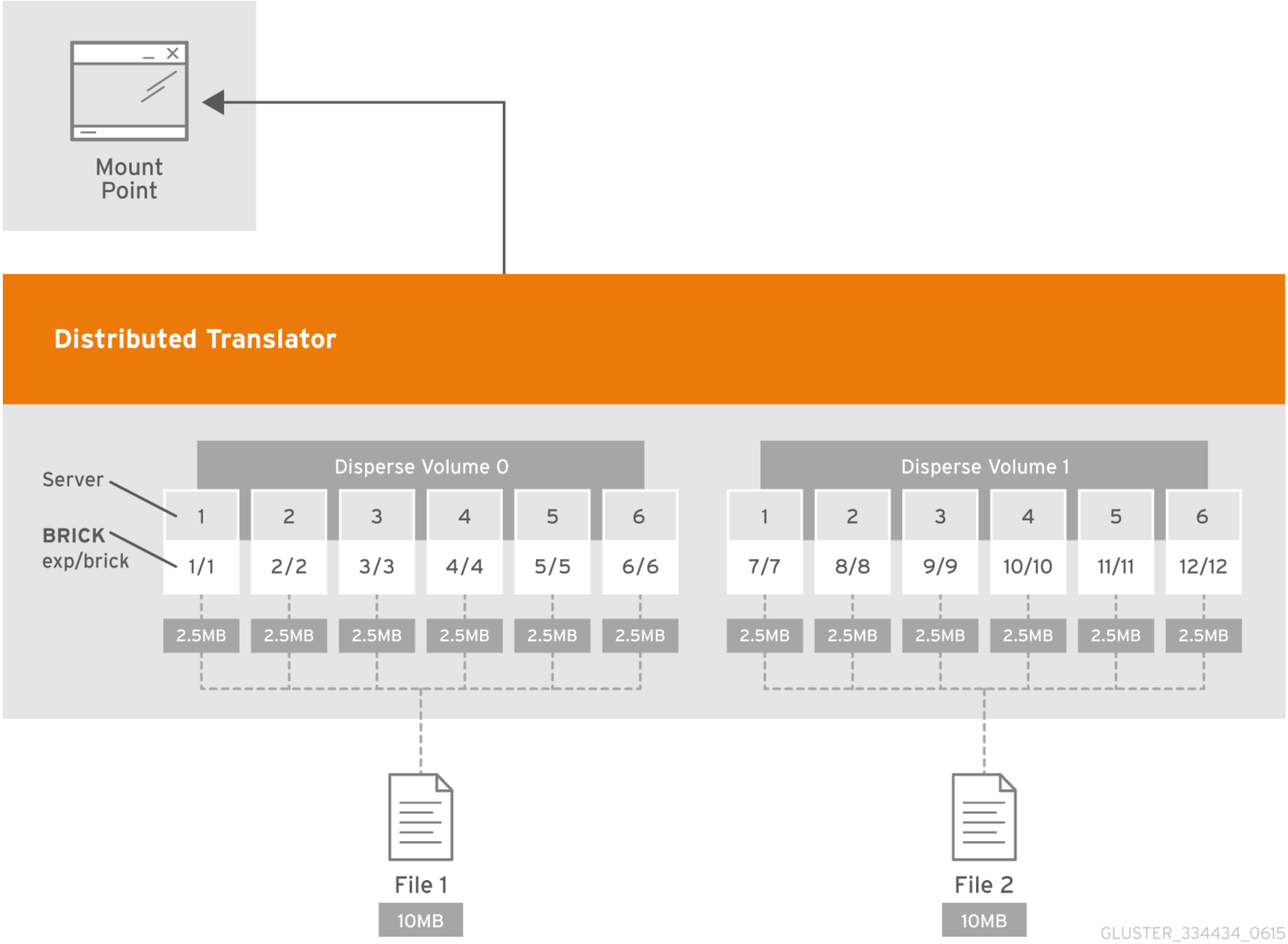


Figure 5.8. Illustration of a Distributed Dispersed Volume

Creating distributed dispersed volumes

Important

Red Hat recommends you to review the Distributed Dispersed Volume configuration recommendations explained in Section 11.16, “Recommended Configurations - Dispersed Volume” before creating the Distributed Dispersed volume.

1. Run the **`gluster volume create`** command to create the dispersed volume.

   The syntax is **`# gluster volume create`** ***`NEW-VOLNAME`*** **`disperse-data`** ***`COUNT`*** **`[redundancy`** ***`COUNT`*** **`] [transport tcp | rdma | tcp,rdma]`** ***`NEW-BRICK...`***

   The default value for transport is **`tcp`**. Other options can be passed such as **`auth.allow`** or **`auth.reject`**. See Section 11.1, “Configuring Volume Options” for a full list of parameters.

   Example 5.12. Distributed Dispersed Volume with Six Storage Servers

   ```
   # gluster volume create test-volume disperse-data 4 redundancy 2
   transport tcp server1:/rhgs1/brick1 server2:/rhgs2/brick2
   server3:/rhgs3/brick3 server4:/rhgs4/brick4 server5:/rhgs5/brick5
   server6:/rhgs6/brick6 server1:/rhgs7/brick7 server2:/rhgs8/brick8
   ```

```
server3:/rhgs9/brick9 server4:/rhgs10/brick10
server5:/rhgs11/brick11 server6:/rhgs12/brick12
Creation of test-volume has been successful
Please start the volume to access data.
```

The above example is illustrated in Figure 5.7, “Illustration of a Dispersed Volume” . In the illustration and example, you are creating 12 bricks from 6 servers.

2. Run **# gluster volume start *VOLNAME*** to start the volume.

```
# gluster volume start test-volume
Starting test-volume has been successful
```

Important

The **open-behind** volume option is enabled by default. If you are accessing the distributed dispersed volume using the SMB protocol, you must disable the **open-behind** volume option to avoid performance bottleneck on large file workload. Run the following command to disable **open-behind** volume option:

```
# gluster volume set VOLNAME open-behind off
```

For information on **open-behind** volume option, see Section 11.1, “Configuring Volume Options”

3. Run **gluster volume info** command to optionally display the volume information.

5.11. Starting Volumes

Volumes must be started before they can be mounted.

To start a volume, run **# gluster volume start *VOLNAME***

Note

Every volume that is created is exported by default through the SMB protocol. If you want to disable it, please refer Section 6.3.7, “Disabling SMB Shares” before starting the volume.

For example, to start test-volume:

```
# gluster volume start test-volume
Starting test-volume has been successful
```

Chapter 6. Creating Access to Volumes

Red Hat Gluster Storage volumes can be accessed using a number of technologies:

- Native Client (see Section 6.1, “Native Client”)
- Network File System (NFS) v3 (see Section 6.2, “NFS”)
- Server Message Block (SMB) (see Section 6.3, “SMB”)

Cross Protocol Data Access

Because of differences in locking semantics, a single Red Hat Gluster Storage volume cannot be concurrently accessed by multiple protocols. Current support for concurrent access is defined in the following table.

Table 6.1. Cross Protocol Data Access Matrix

	SMB	Gluster NFS	NFS-Ganesha	Native FUSE	Object
SMB	Yes	No	No	No	No
Gluster NFS	No	Yes	No	No	No
NFS-Ganesha	No	No	Yes	No	No
Native FUSE	No	No	No	Yes	Yes [a]
Object	No	No	No	Yes [a]	Yes

[a] For more information, refer Section 6.5, “Managing Object Store”.

Access Protocols Supportability

The following table provides the support matrix for the supported access protocols with TCP/RDMA.

Table 6.2. Access Protocol Supportability Matrix

Access Protocols	TCP	RDMA
FUSE	Yes	Yes
SMB	Yes	No
NFS	Yes	Yes

Important

Red Hat Gluster Storage requires certain ports to be open. You must ensure that the firewall settings allow access to the ports listed at Chapter 3, *Considerations for Red Hat Gluster Storage*.

6.1. Native Client

Native Client is a FUSE-based client running in user space. Native Client is the recommended method for accessing Red Hat Gluster Storage volumes when high concurrency and high write performance is required.

This section introduces Native Client and explains how to install the software on client machines. This section also describes how to mount Red Hat Gluster Storage volumes on clients (both manually and automatically) and how to verify that the Red Hat Gluster Storage volume has mounted successfully.

Table 6.3. Red Hat Gluster Storage Support Matrix

Red Hat Enterprise Linux version	Red Hat Gluster Storage version	Native client version
6.5	3.0	3.0, 2.1*
6.6	3.0.2, 3.0.3, 3.0.4	3.0, 2.1*
6.7	3.1, 3.1.1, 3.1.2	3.1, 3.0, 2.1*
6.8	3.1.3	3.1.3
6.9	3.2, 3.3, 3.3.1	3.3.1, 3.3, 3.2
6.10	3.2, 3.3, 3.4	3.4, 3.3.z
7.1	3.1, 3.1.1	3.1, 3.1.1, 3.0
7.2	3.1.2, 3.1.3	3.1.3, 3.1.1, 3.1.2
7.3	3.1.3, 3.2	3.2, 3.1.3
7.4	3.3, 3.3.1	3.3.1, 3.3
7.5	3.3.1, 3.4	3.3.z, 3.4
7.6	3.3.1, 3.4	3.3.z, 3.4

Warning

If you want to access a volume being provided by a server using Red Hat Gluster Storage 3.1.3 or higher, your client must also be using Red Hat Gluster Storage 3.1.3 or higher. Accessing these volumes from earlier client versions can result in data becoming unavailable and problems with directory operations. This requirement exists because Red Hat Gluster Storage 3.1.3 changed how the Distributed Hash Table works in order to improve directory consistency and remove the effects seen in BZ#1115367 and BZ#1118762.

Warning

The following issues are observed and recorded for Red Hat Gluster Storage 3.2 on RHEL 6.x and 7.x using Native Client 3.1.3:

- `gluster volume heal VOLNAME info` is unresponsive for some volumes. (BZ#1500542)
- Gluster brick process crashes frequently. (BZ#1510725)
- Multiple disconnects on NFS mounts. (BZ#1425740)

Warning

- For Red Hat Gluster Storage 3.4, Red Hat supports Red Hat Gluster Storage 3.3 and 3.4 clients only.
- For Red Hat Gluster Storage 3.2, you need to have Red Hat Gluster Storage 3.2 clients. This version is not compatible with backward versions of the client.

For more information on the release version see, https://access.redhat.com/solutions/543123.

6.1.1. Installing Native Client

After installing the client operating system, register the target system to Red Hat Network and subscribe to the Red Hat Enterprise Linux Server channel.

Important

All clients must be of the same version. Red Hat strongly recommends upgrading the servers before upgrading the clients.

Use the Command Line to Register and Subscribe a System to Red Hat Subscription Management

Register the system using the command line, and subscribe to the correct repositories.

Prerequisites

- Know the user name and password of the Red Hat Subscription Manager account with Red Hat Gluster Storage entitlements.

1. Run the **`subscription-manager register`** command to list the available pools. Select the appropriate pool and enter your Red Hat Subscription Manager user name and password to register the system with Red Hat Subscription Manager.

   ```
   # subscription-manager register
   ```

2. Depending on your client, run one of the following commands to subscribe to the correct repositories.

   - For Red Hat Enterprise Linux 7.x clients:

     ```
     # subscription-manager repos --enable=rhel-7-server-rpms --
     enable=rh-gluster-3-client-for-rhel-7-server-rpms
     ```

     Note

     The following command can also be used, but Red Hat Gluster Storage may deprecate support for this repository in future releases.

     ```
     # subscription-manager repos --enable=rhel-7-server-rh-common-
     rpms
     ```

   - For Red Hat Enterprise Linux 6.1 and later clients:

     ```
     # subscription-manager repos --enable=rhel-6-server-rpms --
     enable=rhel-6-server-rhs-client-1-rpms
     ```

   - For Red Hat Enterprise Linux 5.7 and later clients:

     ```
     # subscription-manager repos --enable=rhel-5-server-rpms --
     enable=rhel-5-server-rhs-client-1-rpms
     ```

For more information, see Section 3.2 Registering from the Command Line in *Using and Configuring Red Hat Subscription Management*.

3. Verify that the system is subscribed to the required repositories.

```
# yum repolist
```

Use the Web Interface to Register and Subscribe a System

Register the system using the web interface, and subscribe to the correct channels.

Prerequisites

- Know the user name and password of the Red Hat Subsrciption Management (RHSM) account with Red Hat Gluster Storage entitlements.

1. Log on to Red Hat Subscription Management (https://access.redhat.com/management).
2. Click the Systems link at the top of the screen.
3. Click the name of the system to which the **Red Hat Gluster Storage Native Client** channel must be appended.
4. Click **Alter Channel Subscriptions** in the **Subscribed Channels** section of the screen.
5. Expand the node for Additional Services Channels for **Red Hat Enterprise Linux 7 for x86_64** or **Red Hat Enterprise Linux 6 for x86_64** or for **Red Hat Enterprise Linux 5 for x86_64** depending on the client platform.
6. Click the **Change Subscriptions** button to finalize the changes.

   When the page refreshes, select the **Details** tab to verify the system is subscribed to the appropriate channels.

Install Native Client Packages

Install Native Client packages from Red Hat Network

Prerequisites

- Use the Command Line to Register and Subscribe a System to Red Hat Subscription Management or
- Use the Web Interface to Register and Subscribe a System

1. Run the **yum install** command to install the native client RPM packages.

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

2. For Red Hat Enterprise 5.x client systems, run the **modprobe** command to load FUSE modules before mounting Red Hat Gluster Storage volumes.

```
# modprobe fuse
```

   For more information on loading modules at boot time, see https://access.redhat.com/knowledge/solutions/47028 .

6.1.2. Upgrading Native Client

Before updating the Native Client, subscribe the clients to the channels mentioned in Section 6.1.1, “Installing Native Client”

Warning

If you want to access a volume being provided by a server using Red Hat Gluster Storage 3.1.3 or higher, your client must also be using Red Hat Gluster Storage 3.1.3 or higher. Accessing these volumes from earlier client versions can result in data becoming unavailable and problems with directory operations. This requirement exists because Red Hat Gluster Storage 3.1.3 changed how the Distributed Hash Table works in order to improve directory consistency and remove the effects seen in BZ#1115367 and BZ#1118762.

1. **Unmount gluster volumes**

   Unmount any gluster volumes prior to upgrading the native client.

   ```
   # umount /mnt/glusterfs
   ```

2. **Upgrade the client**

   Run the **yum update** command to upgrade the native client:

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

3. **Remount gluster volumes**

   Remount volumes as discussed in Section 6.1.3, “Mounting Red Hat Gluster Storage Volumes”.

6.1.3. Mounting Red Hat Gluster Storage Volumes

After installing Native Client, the Red Hat Gluster Storage volumes must be mounted to access data. Three methods are available:

- Section 6.1.3.2, “Mounting Volumes Manually”
- Section 6.1.3.3, “Mounting Volumes Automatically”
- Section 6.1.3.4, “Manually Mounting Sub-directories Using Native Client”

After mounting a volume, test the mounted volume using the procedure described in Section 6.1.3.5, “Testing Mounted Volumes”.

Note

- Clients should be on the same version as the server, and at least on the version immediately previous to the server version. For Red Hat Gluster Storage 3.4, the recommended native client version should either be 3.4 or 3.3.z. For other versions, see Section 6.1, “Native Client”.
- Server names selected during volume creation should be resolvable in the client machine. Use appropriate `/etc/hosts` entries, or a DNS server to resolve server names to IP addresses.
- Internet Protocol Version 6 (IPv6) support is available only for Red Hat Hyperconverged Infrastructure for Virtualization environments and not for Red Hat Gluster Storage standalone environments.

6.1.3.1. Mount Commands and Options

The following options are available when using the **`mount -t glusterfs`** command. All options must be separated with commas.

```
# mount -t glusterfs -o backup-
volfile-servers=volfile_server2:volfile_server3:....
..:volfile_serverN,transport-type tcp,log-level=WARNING,reader-thread-
count=2,log-file=/var/log/gluster.log server1:/test-volume /mnt/glusterfs
```

backup-volfile-servers=<volfile_server2>:<volfile_server3>:...:<volfile_serverN>

List of the backup volfile servers to mount the client. If this option is specified while mounting the fuse client, when the first volfile server fails, the servers specified in **`backup-volfile-servers`** option are used as volfile servers to mount the client until the mount is successful.

Note

This option was earlier specified as **`backupvolfile-server`** which is no longer valid.

log-level

Logs only specified level or higher severity messages in the **`log-file`**.

log-file

Logs the messages in the specified file.

transport-type

Specifies the transport type that FUSE client must use to communicate with bricks. If the volume was created with only one transport type, then that becomes the default when no value is specified. In case of **`tcp,rdma`** volume, tcp is the default.

dump-fuse

This mount option creates dump of fuse traffic between the glusterfs client (fuse userspace server) and the kernel. The interface to mount a glusterfs volume is the standard mount(8) command from the CLI. This feature enables the same in the mount option.

```
# mount -t glusterfs -odump-fuse=filename hostname:/volname mount-path
```

For example,

```
# mount -t glusterfs -odump-fuse=/dumpfile  10.70.43.18:/arbiter
/mnt/arbiter
```

The above command generates a binary file with the name **`dumpfile`**.

Note

The fusedump grows large with time and notably if the client gets a heavy load. So this is not an intended use case to do fusedump during normal usage. It is advised to use this to get a dump from a particular scenario, for diagnostic purposes.

You need to unmount and remount the volume without the fusedump option to stop dumping.

ro

Mounts the file system with read-only permissions.

acl

Enables POSIX Access Control List on mount. See Section 6.4.4, “Checking ACL enablement on a mounted volume” for further information.

background-qlen=*n*

Enables FUSE to handle *n* number of requests to be queued before subsequent requests are denied. Default value of *n* is 64.

enable-ino32

Enables file system to present 32-bit inodes instead of 64-bit inodes.

reader-thread-count=*n*

Enables FUSE to add *n* number of reader threads that can give better I/O performance. Default value of *n* is **`1`**.

lru-limit

This **`mount`** command option clears the inodes from the least recently used (lru) list (which keeps non-referenced inodes) after the inode limit has reached.

For example,

```
# mount -olru-limit=NNNN -t glusterfs hostname:/volname /mnt/mountdir
```

Where *NNNN* is a positive integer. The default value of *NNNN* is 128k (131072) and the recommended value is 20000 and above. If **`0`** is specified as the **`lru-limit`** then it means that no invalidation of inodes from the lru-list.

6.1.3.2. Mounting Volumes Manually

Manually Mount a Red Hat Gluster Storage Volume or Subdirectory

Create a mount point and run the following command as required:

For a Red Hat Gluster Storage Volume

```
mount -t glusterfs HOSTNAME|IPADDRESS:/VOLNAME /MOUNTDIR
```

For a Red Hat Gluster Storage Volume's Subdirectory

```
mount -t glusterfs HOSTNAME|IPADDRESS:/VOLNAME/SUBDIRECTORY /MOUNTDIR
```

> **Note**
>
> The server specified in the mount command is used to fetch the glusterFS configuration volfile, which describes the volume name. The client then communicates directly with the servers mentioned in the volfile (which may not actually include the server used for mount).

1. If a mount point has not yet been created for the volume, run the **mkdir** command to create a mount point.

   ```
   # mkdir /mnt/glusterfs
   ```

2. Run the **mount -t glusterfs** command, using the key in the task summary as a guide.

   a. For a Red Hat Gluster Storage Volume:

   ```
   # mount -t glusterfs server1:/test-volume /mnt/glusterfs
   ```

   b. For a Red Hat Gluster Storage Volume's Subdirectory

   ```
   # mount -t glusterfs server1:/test-volume/sub-dir /mnt/glusterfs
   ```

6.1.3.3. Mounting Volumes Automatically

Volumes can be mounted automatically each time the systems starts.

The server specified in the mount command is used to fetch the glusterFS configuration volfile, which describes the volume name. The client then communicates directly with the servers mentioned in the volfile (which may not actually include the server used for mount).

Mounting a Volume Automatically

Mount a Red Hat Gluster Storage Volume automatically at server start.

1. Open the **/etc/fstab** file in a text editor.

2. Append the following configuration to the **fstab** file:

   For a Red Hat Gluster Storage Volume

   ```
   HOSTNAME|IPADDRESS:/VOLNAME /MOUNTDIR glusterfs
   defaults,_netdev 0 0
   ```

   For a Red Hat Gluster Storage Volume's Subdirectory

```
HOSTNAME|IPADDRESS:/VOLNAME/SUBDIRECTORY /MOUNTDIR glusterfs
defaults,_netdev 0 0
```

Using the example server names, the entry contains the following replaced values.

```
server1:/test-volume /mnt/glusterfs glusterfs defaults,_netdev 0 0
```

OR

```
server1:/test-volume/subdir /mnt/glusterfs glusterfs defaults,_netdev
0 0
```

If you want to specify the transport type then check the following example:

```
server1:/test-volume /mnt/glusterfs glusterfs
defaults,_netdev,transport=tcp 0 0
```

OR

```
server1:/test-volume/sub-dir /mnt/glusterfs glusterfs
defaults,_netdev,transport=tcp 0 0
```

6.1.3.4. Manually Mounting Sub-directories Using Native Client

With Red Hat Gluster Storage 3.4, you can share a single Gluster volume with different clients and they all can mount only a subset of the volume namespace. This feature is similar to the NFS subdirectory mount feature where you can export a subdirectory of an already exported volume. You can also use this feature to restrict full access to any particular volume.

Mounting subdirectories provides the following benefits:

- Provides namespace isolation so that multiple users can access the storage without risking namespace collision with other users.
- Prevents the root file system from becoming full in the event of a mount failure.

You can mount a subdirectory using native client by running either of the following commands:

```
# mount -t glusterfs hostname:/volname/subdir /mount-point
```

OR

```
# mount -t glusterfs hostname:/volname -osubdir-mount=subdir /mount-point
```

For example:

```
# gluster volume set test-vol auth.allow
"/(192.168.10.*|192.168.11.*),/subdir1(192.168.1.*),/subdir2(192.168.8.*)”
```

In the above example:

- The **auth.allow** option allows only the directories specified as the value of the **auth.allow** option to be mounted.

- Each group of auth-allow is separated by a comma (**,**).
- Each group has a directory separated by parentheses, **()**, which contains the valid IP addresses.
- All subdirectories start with **/**, that is, no relative path to a volume, but everything is an absolute path, taking **/** as the root directory of the volume.

> **Note**
>
> By default, the authentication is *****, where any given subdirectory in a volume can be mounted by all clients.

6.1.3.5. Testing Mounted Volumes

Testing Mounted Red Hat Gluster Storage Volumes

Using the command-line, verify the Red Hat Gluster Storage volumes have been successfully mounted. All three commands can be run in the order listed, or used independently to verify a volume has been successfully mounted.

Prerequisites

- Section 6.1.3.3, “Mounting Volumes Automatically”, or
- Section 6.1.3.2, “Mounting Volumes Manually”

1. Run the **mount** command to check whether the volume was successfully mounted.

   ```
   # mount
   server1:/test-volume on /mnt/glusterfs type
   fuse.glusterfs(rw,allow_other,default_permissions,max_read=131072
   ```

   OR

   ```
   # mount
   server1:/test-volume/sub-dir on /mnt/glusterfs type
   fuse.glusterfs(rw,allow_other,default_permissions,max_read=131072
   ```

   If transport option is used while mounting a volume, mount status will have the transport type appended to the volume name. For example, for transport=tcp:

   ```
   # mount
   server1:/test-volume.tcp on /mnt/glusterfs type
   fuse.glusterfs(rw,allow_other,default_permissions,max_read=131072
   ```

   OR

   ```
   # mount
   server1:/test-volume/sub-dir.tcp on /mnt/glusterfs type
   fuse.glusterfs(rw,allow_other,default_permissions,max_read=131072
   ```

2. Run the **df** command to display the aggregated storage space from all the bricks in a volume.

```
# df -h /mnt/glusterfs
Filesystem           Size  Used  Avail  Use%  Mounted on
server1:/test-volume  28T  22T   5.4T   82%   /mnt/glusterfs
```

3. Move to the mount directory using the **cd** command, and list the contents.

```
# cd /mnt/glusterfs
# ls
```

6.2. NFS

Red Hat Gluster Storage has two NFS server implementations, Gluster NFS and NFS-Ganesha. Gluster NFS supports only NFSv3 protocol, however, NFS-Ganesha supports NFSv3 and NFSv4 protocols.

- Section 6.2.1, “Support Matrix”
- Section 6.2.2, “Gluster NFS”
- Section 6.2.3, “NFS Ganesha”

6.2.1. Support Matrix

The following table contains the feature matrix of the NFS support on Red Hat Gluster Storage 3.1 and later:

Table 6.4. NFS Support Matrix

Features	glusterFS NFS (NFSv3)	NFS-Ganesha (NFSv3)	NFS-Ganesha (NFSv4)
Root-squash	Yes	Yes	Yes
Sub-directory exports	Yes	Yes	Yes
Locking	Yes	Yes	Yes
Client based export permissions	Yes	Yes	Yes
Netgroups	Yes	Yes	Yes
Mount protocols	UDP, TCP	UDP, TCP	Only TCP
NFS transport protocols	TCP	UDP, TCP	TCP
AUTH_UNIX	Yes	Yes	Yes
AUTH_NONE	Yes	Yes	Yes
AUTH_KRB	No	Yes	Yes
ACLs	Yes	No	Yes
Delegations	N/A	N/A	No
High availability	Yes (but with certain limitations. For more information see, "Setting up CTDB for NFS")	Yes	Yes
Multi-head	Yes	Yes	Yes
Gluster RDMA volumes	Yes	Not supported	Not supported
DRC	Not supported	Yes	Yes
Dynamic exports	No	Yes	Yes
pseudofs	N/A	N/A	Yes
NFSv4.1	N/A	N/A	Not Supported

Note

- Red Hat does not recommend running NFS-Ganesha with any other NFS servers, such as, kernel-NFS and Gluster NFS servers.
- Only one of NFS-Ganesha, gluster-NFS or kernel-NFS servers can be enabled on a given machine/host as all NFS implementations use the port 2049 and only one can be active at a given time. Hence you must disable kernel-NFS before NFS-Ganesha is started.

6.2.2. Gluster NFS

Linux, and other operating systems that support the NFSv3 standard can use NFS to access the Red Hat Gluster Storage volumes.

Note

From the Red Hat Gluster Storage 3.2 release onwards, Gluster NFS server will be disabled by default for any new volumes that are created. You can restart Gluster NFS server on the new volumes explicitly if needed. This can be done running the “**`mount -t nfs`**” command on the client as below:

On any one of the server node:

```
# gluster volume set VOLNAME nfs.disable off
```

However, existing volumes (using Gluster NFS server) will not be impacted even after upgrade to Red Hat Gluster Storage 3.2 and will have implicit enablement of Gluster NFS server.

Differences in implementation of the NFSv3 standard in operating systems may result in some operational issues. If issues are encountered when using NFSv3, contact Red Hat support to receive more information on Red Hat Gluster Storage client operating system compatibility, and information about known issues affecting NFSv3.

NFS ACL v3 is supported, which allows getfacl and setfacl operations on NFS clients. The following options are provided to configure the Access Control Lists (ACL) in the glusterFS NFS server with the nfs.acl option. For example:

- To set nfs.acl ON, run the following command:

  ```
  # gluster volume set VOLNAME nfs.acl on
  ```

- To set nfs.acl OFF, run the following command:

  ```
  # gluster volume set VOLNAME nfs.acl off
  ```

Note

ACL is ON by default.

Red Hat Gluster Storage includes Network Lock Manager (NLM) v4. NLM protocol allows NFSv3 clients to lock files across the network. NLM is required to make applications running on top of NFSv3 mount points to use the standard fcntl() (POSIX) and flock() (BSD) lock system calls to synchronize access across clients.

This section describes how to use NFS to mount Red Hat Gluster Storage volumes (both manually and automatically) and how to verify that the volume has been mounted successfully.

Important

On Red Hat Enterprise Linux 7, enable the firewall service in the active zones for runtime and permanent mode using the following commands:

To get a list of active zones, run the following command:

```
# firewall-cmd --get-active-zones
```

To allow the firewall service in the active zones, run the following commands:

```
# firewall-cmd --zone=zone_name --add-service=nfs --add-service=rpc-
bind
# firewall-cmd --zone=zone_name --add-service=nfs --add-service=rpc-
bind --permanent
```

- Section 6.2.2.1, “Setting up CTDB for Gluster NFS ”
  - Section 6.2.2.1.1, “Prerequisites”
  - Section 6.2.2.1.2, “Port and Firewall Information for Gluster NFS”
  - Section 6.2.2.1.3, “Configuring CTDB on Red Hat Gluster Storage Server”
- Section 6.2.2.2, “Using Gluster NFS to Mount Red Hat Gluster Storage Volumes”
  - Section 6.2.2.2.1, “Manually Mounting Volumes Using Gluster NFS”
  - Section 6.2.2.2.2, “Automatically Mounting Volumes Using Gluster NFS”
  - Section 6.2.2.2.3, “Automatically Mounting Subdirectories Using NFS”
  - Section 6.2.2.2.4, “Testing Volumes Mounted Using Gluster NFS”
- Section 6.2.2.3, “Troubleshooting Gluster NFS”

6.2.2.1. Setting up CTDB for Gluster NFS

In a replicated volume environment, the CTDB software (Cluster Trivial Database) has to be configured to provide high availability and lock synchronization for Samba shares. CTDB provides high availability by adding virtual IP addresses (VIPs) and a heartbeat service.

When a node in the trusted storage pool fails, CTDB enables a different node to take over the virtual IP addresses that the failed node was hosting. This ensures the IP addresses for the services provided are always available. However, locks are not migrated as part of failover.

Important

On Red Hat Enterprise Linux 7, enable the CTDB firewall service in the active zones for runtime and permanent mode using the below commands:

To get a list of active zones, run the following command:

```
# firewall-cmd --get-active-zones
```

To add ports to the active zones, run the following commands:

```
# firewall-cmd --zone=zone_name --add-port=4379/tcp
# firewall-cmd --zone=zone_name --add-port=4379/tcp  --permanent
```

Note

Amazon Elastic Compute Cloud (EC2) does not support VIPs and is hence not compatible with this solution.

6.2.2.1.1. Prerequisites

Follow these steps before configuring CTDB on a Red Hat Gluster Storage Server:

- If you already have an older version of CTDB (version <= ctdb1.x), then remove CTDB by executing the following command:

  ```
  # yum remove ctdb
  ```

  After removing the older version, proceed with installing the latest CTDB.

  Note

  Ensure that the system is subscribed to the samba channel to get the latest CTDB packages.

- Install CTDB on all the nodes that are used as NFS servers to the latest version using the following command:

  ```
  # yum install ctdb
  ```

- CTDB uses TCP port 4379 by default. Ensure that this port is accessible between the Red Hat Gluster Storage servers.

6.2.2.1.2. Port and Firewall Information for Gluster NFS

On the GNFS-Client machine, configure firewalld to add ports used by statd, nlm and portmapper services by executing the following commands:

```
# firewall-cmd --zone=public --add-port=662/tcp --add-port=662/udp \
    --add-port=32803/tcp --add-port=32769/udp \ --add-port=111/tcp --add-
port=111/udp
```

```
# firewall-cmd --zone=public --add-port=662/tcp --add-port=662/udp \
    --add-port=32803/tcp --add-port=32769/udp \ --add-port=111/tcp --add-
port=111/udp --permanent
```

Execute the following steps on the client machine:

- Edit **`/etc/sysconfig/nfs`** file as mentioned below:

  ```
  # sed -i '/STATD_PORT/s/^#//' /etc/sysconfig/nfs
  ```

- Restart the services:

  - For Red Hat Enterprise Linux 6:

    ```
                          # service nfslock restart
                          # service nfs restart
    ```

  - For Red Hat Enterprise Linux 7:

    ```
                          # systemctl restart nfs-config
                          # systemctl restart rpc-statd
                          # systemctl restart nfs-mountd
                          # systemctl restart nfslock
    ```

6.2.2.1.3. Configuring CTDB on Red Hat Gluster Storage Server

To configure CTDB on Red Hat Gluster Storage server, execute the following steps:

1. Create a replicate volume. This volume will host only a zero byte lock file, hence choose minimal sized bricks. To create a replicate volume run the following command:

   ```
   # gluster volume create volname replica n ipaddress:/brick
   path.......N times
   ```

   where,

   N: The number of nodes that are used as Gluster NFS servers. Each node must host one brick.

   For example:

   ```
   # gluster volume create ctdb replica 3
   10.16.157.75:/rhgs/brick1/ctdb/b1 10.16.157.78:/rhgs/brick1/ctdb/b2
   10.16.157.81:/rhgs/brick1/ctdb/b3
   ```

2. In the following files, replace "all" in the statement META="all" to the newly created volume name

   ```
   /var/lib/glusterd/hooks/1/start/post/S29CTDBsetup.sh
   /var/lib/glusterd/hooks/1/stop/pre/S29CTDB-teardown.sh
   ```

For example:

```
META="all"
  to
META="ctdb"
```

3. Start the volume.

   ```
   # gluster volume start ctdb
   ```

   As part of the start process, the **S29CTDBsetup.sh** script runs on all Red Hat Gluster Storage servers, adds an entry in **/etc/fstab** for the mount, and mounts the volume at **/gluster/lock** on all the nodes with Gluster NFS server. It also enables automatic start of CTDB service on reboot.

   > **Note**
   >
   > When you stop the special CTDB volume, the S29CTDB-teardown.sh script runs on all Red Hat Gluster Storage servers and removes an entry in /etc/fstab for the mount and unmounts the volume at /gluster/lock.

4. Verify if the file /etc/sysconfig/ctdb exists on all the nodes that is used as Gluster NFS server. This file contains Red Hat Gluster Storage recommended CTDB configurations.

5. Create /etc/ctdb/nodes file on all the nodes that is used as Gluster NFS servers and add the IPs of these nodes to the file.

   ```
   10.16.157.0
   10.16.157.3
   10.16.157.6
   ```

   The IPs listed here are the private IPs of NFS servers.

6. On all the nodes that are used as Gluster NFS server which require IP failover, create /etc/ctdb/public_addresses file and add the virtual IPs that CTDB should create to this file. Add these IP address in the following format:

   ```
   <Virtual IP>/<routing prefix><node interface>
   ```

   For example:

   ```
   192.168.1.20/24 eth0
   192.168.1.21/24 eth0
   ```

7. Start the CTDB service on all the nodes by executing the following command:

   ```
   # service ctdb start
   ```

Note

CTDB with gNFS only provides node level high availability and is not capable of detecting NFS service failure. Therefore, CTDB does not provide high availability if the NFS service goes down while the node is still up and running.

6.2.2.2. Using Gluster NFS to Mount Red Hat Gluster Storage Volumes

You can use either of the following methods to mount Red Hat Gluster Storage volumes:

Note

Currently GlusterFS NFS server only supports version 3 of NFS protocol. As a preferred option, always configure version 3 as the default version in the **nfsmount.conf** file at **/etc/nfsmount.conf** by adding the following text in the file:

```
Defaultvers=3
```

In case the file is not modified, then ensure to add **vers=3** manually in all the mount commands.

```
# mount nfsserver:export -o vers=3 /MOUNTPOINT
```

RDMA support in GlusterFS that is mentioned in the previous sections is with respect to communication between bricks and Fuse mount/GFAPI/NFS server. NFS kernel client will still communicate with GlusterFS NFS server over tcp.

In case of volumes which were created with only one type of transport, communication between GlusterFS NFS server and bricks will be over that transport type. In case of **tcp,rdma** volume it could be changed usinc the volume set option **nfs.transport-type**.

- Section 6.2.2.2.1, “Manually Mounting Volumes Using Gluster NFS”
- Section 6.2.2.2.2, “Automatically Mounting Volumes Using Gluster NFS”

After mounting a volume, you can test the mounted volume using the procedure described in .Section 6.2.2.2.4, “Testing Volumes Mounted Using Gluster NFS”

6.2.2.2.1. Manually Mounting Volumes Using Gluster NFS

Create a mount point and run the **mount** command to manually mount a Red Hat Gluster Storage volume using Gluster NFS.

1. If a mount point has not yet been created for the volume, run the **mkdir** command to create a mount point.

   ```
   # mkdir /mnt/glusterfs
   ```

2. Run the correct **mount** command for the system.

   For Linux

   ```
   # mount -t nfs -o vers=3 server1:/test-volume /mnt/glusterfs
   ```

For Solaris

```
# mount -o vers=3 nfs://server1:38467/test-volume
/mnt/glusterfs
```

Manually Mount a Red Hat Gluster Storage Volume using Gluster NFS over TCP

Create a mount point and run the **mount** command to manually mount a Red Hat Gluster Storage volume using Gluster NFS over TCP.

Note

glusterFS NFS server does not support UDP. If a NFS client such as Solaris client, connects by default using UDP, the following message appears:

requested NFS version or transport protocol is not supported

The option **nfs.mount-udp** is supported for mounting a volume, by default it is disabled. The following are the limitations:

- If **nfs.mount-udp** is enabled, the MOUNT protocol needed for NFSv3 can handle requests from NFS-clients that require MOUNT over UDP. This is useful for at least some versions of Solaris, IBM AIX and HP-UX.
- Currently, MOUNT over UDP does not have support for mounting subdirectories on a volume. Mounting **server:/volume/subdir** exports is only functional when MOUNT over TCP is used.
- MOUNT over UDP does not currently have support for different authentication options that MOUNT over TCP honors. Enabling **nfs.mount-udp** may give more permissions to NFS clients than intended via various authentication options like **nfs.rpc-auth-allow**, **nfs.rpc-auth-reject** and **nfs.export-dir**.

1. If a mount point has not yet been created for the volume, run the **mkdir** command to create a mount point.

```
# mkdir /mnt/glusterfs
```

2. Run the correct **mount** command for the system, specifying the TCP protocol option for the system.

For Linux

```
# mount -t nfs -o vers=3,mountproto=tcp server1:/test-volume
/mnt/glusterfs
```

For Solaris

```
# mount -o proto=tcp, nfs://server1:38467/test-volume
/mnt/glusterfs
```

6.2.2.2.2. Automatically Mounting Volumes Using Gluster NFS

Red Hat Gluster Storage volumes can be mounted automatically using Gluster NFS, each time the system starts.

Note

In addition to the tasks described below, Red Hat Gluster Storage supports Linux, UNIX, and similar operating system's standard method of auto-mounting Gluster NFS mounts.

Update the **`/etc/auto.master`** and **`/etc/auto.misc`** files, and restart the **`autofs`** service. Whenever a user or process attempts to access the directory it will be mounted in the background on-demand.

Mounting a Volume Automatically using NFS

Mount a Red Hat Gluster Storage Volume automatically using NFS at server start.

1. Open the **`/etc/fstab`** file in a text editor.
2. Append the following configuration to the **`fstab`** file.

```
HOSTNAME|IPADDRESS:/VOLNAME /MOUNTDIR nfs defaults,_netdev, 0 0
```

Using the example server names, the entry contains the following replaced values.

```
server1:/test-volume /mnt/glusterfs nfs defaults,_netdev, 0 0
```

Mounting a Volume Automatically using NFS over TCP

Mount a Red Hat Gluster Storage Volume automatically using NFS over TCP at server start.

1. Open the **`/etc/fstab`** file in a text editor.
2. Append the following configuration to the **`fstab`** file.

```
HOSTNAME|IPADDRESS:/VOLNAME /MOUNTDIR nfs
defaults,_netdev,mountproto=tcp 0 0
```

Using the example server names, the entry contains the following replaced values.

```
server1:/test-volume /mnt/glusterfs nfs
defaults,_netdev,mountproto=tcp 0 0
```

6.2.2.2.3. Automatically Mounting Subdirectories Using NFS

The **`nfs.export-dir`** and **`nfs.export-dirs`** options provide granular control to restrict or allow specific clients to mount a sub-directory. These clients can be authenticated during sub-directory mount with either an IP, host name or a Classless Inter-Domain Routing (CIDR) range.

nfs.export-dirs

This option is enabled by default. It allows the sub-directories of exported volumes to be mounted by clients without needing to export individual sub-directories. When enabled, all sub-directories of all volumes are exported. When disabled, sub-directories must be exported individually in order to mount them on clients.

To disable this option for all volumes, run the following command:

```
# gluster volume set VOLNAME nfs.export-dirs off
```

nfs.export-dir

When **`nfs.export-dirs`** is set to **on**, the **`nfs.export-dir`** option allows you to specify one or more sub-directories to export, rather than exporting all subdirectories (**`nfs.export-dirs on`**), or only exporting individually exported subdirectories (**`nfs.export-dirs off`**).

To export certain subdirectories, run the following command:

```
# gluster volume set VOLNAME nfs.export-dir subdirectory
```

The subdirectory path should be the path from the root of the volume. For example, in a volume with six subdirectories, to export the first three subdirectories, the command would be the following:

```
# gluster volume set myvolume nfs.export-dir /dir1,/dir2,/dir3
```

Subdirectories can also be exported based on the IP address, hostname, or a Classless Inter-Domain Routing (CIDR) range by adding these details in parentheses after the directory path:

```
# gluster volume set VOLNAME nfs.export-dir
subdirectory(IPADDRESS),subdirectory(HOSTNAME),subdirectory(CIDR)
```

```
# gluster volume set myvolume nfs.export-dir
/dir1(192.168.10.101),/dir2(storage.example.com),/dir3(192.168.98.0/2
4)
```

6.2.2.2.4. Testing Volumes Mounted Using Gluster NFS

You can confirm that Red Hat Gluster Storage directories are mounting successfully.

To test mounted volumes

Testing Mounted Red Hat Gluster Storage Volumes

Using the command-line, verify the Red Hat Gluster Storage volumes have been successfully mounted. All three commands can be run in the order listed, or used independently to verify a volume has been successfully mounted.

Prerequisites

- Section 6.2.2.2.2, “Automatically Mounting Volumes Using Gluster NFS” , or
- Section 6.2.2.2.1, “Manually Mounting Volumes Using Gluster NFS”

1. Run the **`mount`** command to check whether the volume was successfully mounted.

```
# mount
server1:/test-volume on /mnt/glusterfs type nfs (rw,addr=server1)
```

2. Run the **`df`** command to display the aggregated storage space from all the bricks in a volume.

```
# df -h /mnt/glusterfs
Filesystem              Size Used Avail Use% Mounted on
server1:/test-volume    28T  22T  5.4T  82%  /mnt/glusterfs
```

3. Move to the mount directory using the **cd** command, and list the contents.

```
# cd /mnt/glusterfs
# ls
```

6.2.2.3. Troubleshooting Gluster NFS

Q: **The mount command on the NFS client fails with `RPC Error: Program not registered`. This error is encountered due to one of the following reasons:**

- **The NFS server is not running. You can check the status using the following command:**

  ```
  # gluster volume status
  ```

- **The volume is not started. You can check the status using the following command:**

  ```
  # gluster volume info
  ```

- **rpcbind is restarted. To check if rpcbind is running, execute the following command:**

  ```
  # ps ax| grep rpcbind
  ```

A:

- If the NFS server is not running, then restart the NFS server using the following command:

  ```
  # gluster volume start VOLNAME
  ```

- If the volume is not started, then start the volume using the following command:

  ```
  # gluster volume start VOLNAME
  ```

- If both rpcbind and NFS server is running then restart the NFS server using the following commands:

  ```
  # gluster volume stop VOLNAME
  ```

  ```
  # gluster volume start VOLNAME
  ```

Q: **The `rpcbind` service is not running on the NFS client. This could be due to the following reasons:**

- **The portmap is not running.**
- **Another instance of kernel NFS server or glusterNFS server is running.**

A: Start the **`rpcbind`** service by running the following command:

```
# service rpcbind start
```

Q: **The NFS server glusterfsd starts but the initialization fails with *nfsrpc- service: portmap registration of program failed* error message in the log.**

A: NFS start-up succeeds but the initialization of the NFS service can still fail preventing clients from accessing the mount points. Such a situation can be confirmed from the following error messages in the log file:

```
[2010-05-26 23:33:47] E [rpcsvc.c:2598:rpcsvc_program_register_portmap]
rpc-service: Could notregister with portmap
[2010-05-26 23:33:47] E [rpcsvc.c:2682:rpcsvc_program_register] rpc-
service: portmap registration of program failed
[2010-05-26 23:33:47] E [rpcsvc.c:2695:rpcsvc_program_register] rpc-
service: Program registration failed: MOUNT3, Num: 100005, Ver: 3, Port:
38465
[2010-05-26 23:33:47] E [nfs.c:125:nfs_init_versions] nfs: Program init
failed
[2010-05-26 23:33:47] C [nfs.c:531:notify] nfs: Failed to initialize
protocols
[2010-05-26 23:33:49] E
[rpcsvc.c:2614:rpcsvc_program_unregister_portmap] rpc-service: Could not
unregister with portmap
[2010-05-26 23:33:49] E [rpcsvc.c:2731:rpcsvc_program_unregister] rpc-
service: portmap unregistration of program failed
[2010-05-26 23:33:49] E [rpcsvc.c:2744:rpcsvc_program_unregister] rpc-
service: Program unregistration failed: MOUNT3, Num: 100005, Ver: 3,
Port: 38465
```

1. Start the rpcbind service on the NFS server by running the following command:

   ```
   # service rpcbind start
   ```

   After starting rpcbind service, glusterFS NFS server needs to be restarted.

2. Stop another NFS server running on the same machine.

   Such an error is also seen when there is another NFS server running on the same machine but it is not the glusterFS NFS server. On Linux systems, this could be the kernel NFS server. Resolution involves stopping the other NFS server or not running the glusterFS NFS server on the machine. Before stopping the kernel NFS server, ensure that no critical service depends on access to that NFS server's exports.

   On Linux, kernel NFS servers can be stopped by using either of the following commands depending on the distribution in use:

   ```
   # service nfs-kernel-server stop
   # service nfs stop
   ```

3. Restart glusterFS NFS server.

Q: The NFS server start-up fails with the message *Port is already in use* in the log file.

A: This error can arise in case there is already a glusterFS NFS server running on the same machine. This situation can be confirmed from the log file, if the following error lines exist:

```
[2010-05-26 23:40:49] E [rpc-socket.c:126:rpcsvc_socket_listen] rpc-
socket: binding socket failed:Address already in use
[2010-05-26 23:40:49] E [rpc-socket.c:129:rpcsvc_socket_listen] rpc-
socket: Port is already in use
[2010-05-26 23:40:49] E [rpcsvc.c:2636:rpcsvc_stage_program_register]
rpc-service: could not create listening connection
```

```
[2010-05-26 23:40:49] E [rpcsvc.c:2675:rpcsvc_program_register] rpc-
service: stage registration of program failed
[2010-05-26 23:40:49] E [rpcsvc.c:2695:rpcsvc_program_register] rpc-
service: Program registration failed: MOUNT3, Num: 100005, Ver: 3, Port:
38465
[2010-05-26 23:40:49] E [nfs.c:125:nfs_init_versions] nfs: Program init
failed
[2010-05-26 23:40:49] C [nfs.c:531:notify] nfs: Failed to initialize
protocols
```

In this release, the glusterFS NFS server does not support running multiple NFS servers on the same machine. To resolve the issue, one of the glusterFS NFS servers must be shutdown.

Q: The `mount` command fails with NFS server failed error:

A:

```
mount: mount to NFS server '10.1.10.11' failed: timed out (retrying).
```

Review and apply the suggested solutions to correct the issue.

- Disable name lookup requests from NFS server to a DNS server.

  The NFS server attempts to authenticate NFS clients by performing a reverse DNS lookup to match host names in the volume file with the client IP addresses. There can be a situation where the NFS server either is not able to connect to the DNS server or the DNS server is taking too long to respond to DNS request. These delays can result in delayed replies from the NFS server to the NFS client resulting in the timeout error.

  NFS server provides a work-around that disables DNS requests, instead relying only on the client IP addresses for authentication. The following option can be added for successful mounting in such situations:

  ```
  option nfs.addr.namelookup off
  ```

  > **Note**
  >
  > Remember that disabling the NFS server forces authentication of clients to use only IP addresses. If the authentication rules in the volume file use host names, those authentication rules will fail and client mounting will fail.

- NFS version used by the NFS client is other than version 3 by default.

  glusterFS NFS server supports version 3 of NFS protocol by default. In recent Linux kernels, the default NFS version has been changed from 3 to 4. It is possible that the client machine is unable to connect to the glusterFS NFS server because it is using version 4 messages which are not understood by glusterFS NFS server. The timeout can be resolved by forcing the NFS client to use version 3. The **vers** option to mount command is used for this purpose:

  ```
  # mount nfsserver:export -o vers=3 /MOUNTPOINT
  ```

Q: The showmount command fails with *clnt_create: RPC: Unable to receive* error. This error is encountered due to the following reasons:

- **The firewall might have blocked the port.**

- **rpcbind might not be running.**

A: Check the firewall settings, and open ports 111 for portmap requests/replies and glusterFS NFS server requests/replies. glusterFS NFS server operates over the following port numbers: 38465, 38466, and 38467.

Q: **The application fails with *Invalid argument* or *Value too large for defined data type***

A: These two errors generally happen for 32-bit NFS clients, or applications that do not support 64-bit inode numbers or large files.

Use the following option from the command-line interface to make glusterFS NFS return 32-bit inode numbers instead:

```
NFS.enable-ino32 <on | off>
```

This option is **`off`** by default, which permits NFS to return 64-bit inode numbers by default.

Applications that will benefit from this option include those that are:

- built and run on 32-bit machines, which do not support large files by default,
- built to 32-bit standards on 64-bit systems.

Applications which can be rebuilt from source are recommended to be rebuilt using the following flag with gcc:

```
-D_FILE_OFFSET_BITS=64
```

Q: **After the machine that is running NFS server is restarted the client fails to reclaim the locks held earlier.**

A: The Network Status Monitor (NSM) service daemon (rpc.statd) is started before gluster NFS server. Hence, NSM sends a notification to the client to reclaim the locks. When the clients send the reclaim request, the NFS server does not respond as it is not started yet. Hence the client request fails.

Solution: To resolve the issue, prevent the NSM daemon from starting when the server starts.

Run **`chkconfig --list nfslock`** to check if NSM is configured during OS boot.

If any of the entries are **`on,`** run **`chkconfig nfslock off`** to disable NSM clients during boot, which resolves the issue.

Q: **The `rpc actor failed to complete successfully` error is displayed in the nfs.log, even after the volume is mounted successfully.**

A: gluster NFS supports only NFS version 3. When nfs-utils mounts a client when the version is not mentioned, it tries to negotiate using version 4 before falling back to version 3. This is the cause of the messages in both the server log and the **`nfs.log`** file.

```
[2013-06-25 00:03:38.160547] W [rpcsvc.c:180:rpcsvc_program_actor] 0-
rpc-service: RPC program version not available (req 100003 4)
[2013-06-25 00:03:38.160669] E
[rpcsvc.c:448:rpcsvc_check_and_reply_error] 0-rpcsvc: rpc actor failed
to complete successfully
```

To resolve the issue, declare NFS version 3 and the **`noacl`** option in the mount command as follows:

```
# mount -t nfs -o vers=3,noacl server1:/test-volume /mnt/glusterfs
```

Q: **The mount command fails with `No such file or directory`.**

A: This problem is encountered as the volume is not present.

6.2.3. NFS Ganesha

NFS-Ganesha is a user space file server for the NFS protocol with support for NFSv3 and NFSv4.

Red Hat Gluster Storage 3.4 is supported with the community's V2.5 stable release of NFS-Ganesha on Red Hat Enterprise Linux 7. To understand the various supported features of NFS-ganesha see, *Supported Features of NFS-Ganesha*.

Note

To install NFS-Ganesha refer, *Deploying NFS-Ganesha on Red Hat Gluster Storage* in the *Red Hat Gluster Storage 3.4 Installation Guide*.

Red Hat Gluster Storage does not support NFSv4 delegations. For more information refer, *Support matrix*.

- Section 6.2.3.1, "Supported Features of NFS-Ganesha"
- Section 6.2.3.2, "Setting up NFS Ganesha"
  - Section 6.2.3.2.1, "Port and Firewall Information for NFS-Ganesha"
  - Section 6.2.3.2.2, "Prerequisites to run NFS-Ganesha"
  - Section 6.2.3.2.3, "Configuring the Cluster Services"
  - Section 6.2.3.2.4, "Creating the ganesha-ha.conf file"
  - Section 6.2.3.2.5, "Configuring NFS-Ganesha using Gluster CLI"
  - Section 6.2.3.2.6, "Exporting and Unexporting Volumes through NFS-Ganesha"
  - Section 6.2.3.2.7, "Verifying the NFS-Ganesha Status"
- Section 6.2.3.3, "Accessing NFS-Ganesha Exports"
  - Section 6.2.3.3.1, "Mounting exports in NFSv3 Mode"
  - Section 6.2.3.3.2, "Mounting exports in NFSv4 Mode"
  - Section 6.2.3.3.3, "Finding clients of an NFS server using dbus"
- Section 6.2.3.4, "Modifying the NFS-Ganesha HA Setup"
  - Section 6.2.3.4.1, "Adding a Node to the Cluster"
  - Section 6.2.3.4.2, "Deleting a Node in the Cluster"

- Section 6.2.3.5, “Modifying the Default Export Configurations”
  - Section 6.2.3.5.1, “Providing Permissions for Specific Clients”
  - Section 6.2.3.5.2, “Enabling and Disabling NFSv4 ACLs ”
  - Section 6.2.3.5.3, “Providing Pseudo Path for NFSv4 Mount”
  - Section 6.2.3.5.4, “Exporting Subdirectories”
  - Section 6.2.3.5.5, “Configuring Upcall Poll Interval”
- Section 6.2.3.6, “Configuring Kerberized NFS-Ganesha”
  - Section 6.2.3.6.1, “Setting up the NFS-Ganesha Server:”
  - Section 6.2.3.6.2, “Setting up the NFS Client”
- Section 6.2.3.7, “NFS-Ganesha Service Downtime”
  - Section 6.2.3.7.1, “Modifying the Fail-over Time”
- Section 6.2.3.8, “Manually Configuring NFS-Ganesha Exports”
- Section 6.2.3.9, “Troubleshooting”

6.2.3.1. Supported Features of NFS-Ganesha

The following list briefly describes the supported features of NFS-Ganesha:

Highly Available Active-Active NFS-Ganesha

In a highly available active-active environment, if a NFS-Ganesha server that is connected to a NFS client running a particular application goes down, the application/NFS client is seamlessly connected to another NFS-Ganesha server without any administrative intervention.

Data coherency across the multi-head NFS-Ganesha servers in the cluster is achieved using the Gluster’s Upcall infrastructure. Gluster’s Upcall infrastructure is a generic and extensible framework that sends notifications to the respective glusterfs clients (in this case NFS-Ganesha server) when changes are detected in the back-end file system.

Dynamic Export of Volumes

NFS-Ganesha supports addition and removal of exports dynamically. Dynamic exports is managed by the DBus interface. DBus is a system local IPC mechanism for system management and peer-to-peer application communication.

Exporting Multiple Entries

In NFS-Ganesha, multiple Red Hat Gluster Storage volumes or sub-directories can be exported simultaneously.

Pseudo File System

NFS-Ganesha creates and maintains a NFSv4 pseudo-file system, which provides clients with seamless access to all exported objects on the server.

Access Control List

NFS-Ganesha NFSv4 protocol includes integrated support for Access Control List (ACL)s, which are similar to those used by Windows. These ACLs can be used to identify a trustee and specify the access rights allowed, or denied for that trustee.This feature is disabled by default.

Note

AUDIT and ALARM ACE types are not currently supported.

6.2.3.2. Setting up NFS Ganesha

To set up NFS Ganesha, follow the steps mentioned in the further sections.

Note

You can also set up NFS-Ganesha using gdeploy, that automates the steps mentioned below. For more information, see "Deploying NFS-Ganesha"

6.2.3.2.1. Port and Firewall Information for NFS-Ganesha

You must ensure to open the ports and firewall services:

The following table lists the port details for NFS-Ganesha cluster setup:

Table 6.5. NFS Port Details

Service	Port Number	Protocol
sshd	22	TCP
rpcbind/portmapper	111	TCP/UDP
NFS	2049	TCP/UDP
mountd	20048	TCP/UDP
NLM	32803	TCP/UDP
RQuota	875	TCP/UDP
statd	662	TCP/UDP
pcsd	2224	TCP
pacemaker_remote	3121	TCP
corosync	5404 and 5405	UDP
dlm	21064	TCP

Note

The port details for the Red Hat Gluster Storage services are listed under section *3. Verifying Port Access*.

Defining Service Ports

Ensure the statd service is configured to use the ports mentioned above by executing the following commands on every node in the nfs-ganesha cluster:

1. Edit /etc/sysconfig/nfs file as mentioned below:

   ```
   # sed -i '/STATD_PORT/s/^#//' /etc/sysconfig/nfs
   ```

2. Restart the statd service:

   For Red Hat Enterprise Linux 7:

   ```
   # systemctl restart nfs-config
   # systemctl restart rpc-statd
   ```

Note

For the NFS client to use the LOCK functionality, the ports used by LOCKD and STATD daemons has to be configured and opened via firewalld on the client machine:

1. Edit '/etc/sysconfig/nfs' using following commands:

   ```
   # sed -i '/STATD_PORT/s/^#//' /etc/sysconfig/nfs
   # sed -i '/LOCKD_TCPPORT/s/^#//' /etc/sysconfig/nfs
   # sed -i '/LOCKD_UDPPORT/s/^#//' /etc/sysconfig/nfs
   ```

2. Restart the services:

   For Red Hat Enterprise Linux 7:

   ```
   # systemctl restart nfs-config
   # systemctl restart rpc-statd
   # systemctl restart nfslock
   ```

3. Open the ports that are configured in the first step using the following commnad:

   ```
   # firewall-cmd --zone=zone_name --add-port=662/tcp --add-
   port=662/udp \
   --add-port=32803/tcp --add-port=32769/udp

   # firewall-cmd --zone=zone_name --add-port=662/tcp --add-
   port=662/udp \
   --add-port=32803/tcp --add-port=32769/udp --permanent
   ```

4. To ensure NFS client UDP mount does not fail, ensure to open port 2049 by executing the following command:

   ```
   # firewall-cmd --zone=zone_name --add-port=2049/udp
   # firewall-cmd --zone=zone_name --add-port=2049/udp --permanent
   ```

- **Firewall Settings**

  On Red Hat Enterprise Linux 7, enable the firewall services mentioned below.

  - Get a list of active zones using the following command:

```
# firewall-cmd --get-active-zones
```

- Allow the firewall service in the active zones, run the following commands:

```
# firewall-cmd --zone=zone_name --add-service=nlm  --add-service=nfs
--add-service=rpc-bind  --add-service=high-availability --add-
service=mountd --add-service=rquota

# firewall-cmd --zone=zone_name  --add-service=nlm  --add-
service=nfs  --add-service=rpc-bind  --add-service=high-availability
--add-service=mountd --add-service=rquota --permanent

# firewall-cmd --zone=zone_name --add-port=662/tcp --add-
port=662/udp

# firewall-cmd --zone=zone_name --add-port=662/tcp --add-
port=662/udp --permanent
```

6.2.3.2.2. Prerequisites to run NFS-Ganesha

Ensure that the following prerequisites are taken into consideration before you run NFS-Ganesha in your environment:

- A Red Hat Gluster Storage volume must be available for export and NFS-Ganesha rpms are installed.

- Ensure that the fencing agents are configured. For more information on configuring fencing agents, refer to the following documenation:

  - Fencing Configuration section in the High Availability Add-On Administration guide: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/high_availability_add-on_administration/s1-fenceconfig-haaa

  - Fence Devices section in the High Availability Add-On Reference guide: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/high_availability_add-on_reference/s1-guiclustcomponents-haar#s2-guifencedevices-HAAR

- Only one of NFS-Ganesha, gluster-NFS or kernel-NFS servers can be enabled on a given machine/host as all NFS implementations use the port 2049 and only one can be active at a given time. Hence you must disable kernel-NFS before NFS-Ganesha is started.

  Disable the kernel-nfs using the following command:

  For Red Hat Enterprise Linux 7

  ```
  # systemctl stop nfs-server
  # systemctl disable nfs-server
  ```

  To verify if kernel-nfs is disabled, execute the following command:

  ```
  # systemctl status nfs-server
  ```

  The service should be in stopped state.

Note

Gluster NFS will be stopped automatically when NFS-Ganesha is enabled.

Ensure that none of the volumes have the variable **nfs.disable** set to 'off'.

- Ensure to configure the ports as mentioned in *Port/Firewall Information for NFS-Ganesha*.
- Edit the ganesha-ha.conf file based on your environment.
- Reserve virtual IPs on the network for each of the servers configured in the ganesha.conf file. Ensure that these IPs are different than the hosts' static IPs and are not used anywhere else in the trusted storage pool or in the subnet.
- Ensure that all the nodes in the cluster are DNS resolvable. For example, you can populate the /etc/hosts with the details of all the nodes in the cluster.
- Make sure the SELinux is in **Enforcing** mode.
- On Red Hat Enterprise Linux 7, execute the following commands to disable and stop NetworkManager service and to enable the network service.

```
# systemctl disable NetworkManager
# systemctl stop NetworkManager
# systemctl enable network
```

- Start network service on all machines using the following command:

  For Red Hat Enterprise Linux 7:

```
# systemctl start network
```

- Create and mount a gluster shared volume by executing the following command:

```
# gluster volume set all cluster.enable-shared-storage enable
volume set: success
```

  For more information, see Section 11.12, “Setting up Shared Storage Volume”

- Create a directory named **nfs-ganesha** under **/var/run/gluster/shared_storage**
- Copy the **ganesha.conf** and **ganesha-ha.conf** files from **/etc/ganesha** to **/var/run/gluster/shared_storage/nfs-ganesha**.
- Enable the glusterfssharedstorage.service service using the following command:

```
systemctl enable glusterfssharedstorage.service
```

- Enable the nfs-ganesha service using the following command:

```
systemctl enable nfs-ganesha
```

6.2.3.2.3. Configuring the Cluster Services

The HA cluster is maintained using Pacemaker and Corosync. Pacemaker acts a resource manager and Corosync provides the communication layer of the cluster. For more information about Pacemaker/Corosync see the documentation under the *Clustering* section of the Red Hat Enterprise Linux 7 documentation: https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/

Note

It is recommended to use 3 or more nodes to configure NFS Ganesha HA cluster, in order to maintain cluster quorum.

1. Enable the pacemaker service using the following command:

   For Red Hat Enterprise Linux 7:

   ```
   # systemctl enable pacemaker.service
   ```

2. Start the pcsd service using the following command.

   For Red Hat Enterprise Linux 7:

   ```
   # systemctl start pcsd
   ```

   ### Note

   - To start pcsd by default after the system is rebooted, execute the following command:

     For Red Hat Enterprise Linux 7:

     ```
     # systemctl enable pcsd
     ```

3. Set a password for the user ‘hacluster’ on all the nodes using the following command. Use the same password for all the nodes:

   ```
   # echo <password> | passwd --stdin hacluster
   ```

4. Perform cluster authentication between the nodes, where, username is ‘hacluster’, and password is the one you used in the previous step. Ensure to execute the following command on every node:

   ```
   # pcs cluster auth <hostname1> <hostname2> ...
   ```

   ### Note

   The hostname of all the nodes in the Ganesha-HA cluster must be included in the command when executing it on every node.

   For example, in a four node cluster; nfs1, nfs2, nfs3, and nfs4, execute the following command on every node:

```
# pcs cluster auth nfs1 nfs2 nfs3 nfs4
Username: hacluster
Password:
nfs1: Authorized
nfs2: Authorized
nfs3: Authorized
nfs4: Authorized
```

5. Key-based SSH authentication without password for the root user has to be enabled on all the HA nodes. Follow these steps:

   a. On one of the nodes (node1) in the cluster, run:

   ```
   # ssh-keygen -f /var/lib/glusterd/nfs/secret.pem -t rsa -N ''
   ```

   b. Deploy the generated public key from node1 to all the nodes (including node1) by executing the following command for every node:

   ```
   # ssh-copy-id -i /var/lib/glusterd/nfs/secret.pem.pub
   root@<node-ip/hostname>
   ```

   c. Copy the ssh keypair from node1 to all the nodes in the Ganesha-HA cluster by executing the following command for every node:

   ```
   # scp -i /var/lib/glusterd/nfs/secret.pem
   /var/lib/glusterd/nfs/secret.* root@<node-
   ip/hostname>:/var/lib/glusterd/nfs/
   ```

6. As part of cluster setup, port 875 is used to bind to the Rquota service. If this port is already in use, assign a different port to this service by modifying following line in '/etc/ganesha/ganesha.conf' file on all the nodes.

```
# Use a non-privileged port for RQuota
Rquota_Port = 875;
```

6.2.3.2.4. Creating the ganesha-ha.conf file

The ganesha-ha.conf.sample is created in the following location /etc/ganesha when Red Hat Gluster Storage is installed. Rename the file to ganesha-ha.conf and make the changes based on your environment.

1. Create a directory named nfs-ganesha under /var/run/gluster/shared_storage

2. Copy the ganesha.conf and ganesha-ha.conf files from /etc/ganesha to /var/run/gluster/shared_storage/nfs-ganesha.

Sample ganesha-ha.conf file:

```
# Name of the HA cluster created.
# must be unique within the subnet
HA_NAME="ganesha-ha-360"
#
#
# You may use short names or long names; you may not use IP addresses.
# Once you select one, stay with it as it will be mildly unpleasant to clean
```

```
# up if you switch later on. Ensure that all names - short and/or long - are
in
# DNS or /etc/hosts on all machines in the cluster.
#
# The subset of nodes of the Gluster Trusted Pool that form the ganesha HA
# cluster. Hostname is specified.
HA_CLUSTER_NODES="server1.lab.redhat.com,server2.lab.redhat.com,..."
#
# Virtual IPs for each of the nodes specified above.
VIP_server1="10.0.2.1"
VIP_server2="10.0.2.2"
#VIP_server1_lab_redhat_com="10.0.2.1"
#VIP_server2_lab_redhat_com="10.0.2.2"
....
....
```

Note

- Pacemaker handles the creation of the VIP and assigning an interface.
- Ensure that the VIP is in the same network range.
- Ensure that the HA_CLUSTER_NODES are specified as hostnames. Using IP addresses will cause clustering to fail.

6.2.3.2.5. Configuring NFS-Ganesha using Gluster CLI

Setting up the HA cluster

To setup the HA cluster, enable NFS-Ganesha by executing the following command:

1. Enable NFS-Ganesha by executing the following command

   ```
   # gluster nfs-ganesha enable
   ```

   Note

   Before enabling or disabling NFS-Ganesha, ensure that all the nodes that are part of the NFS-Ganesha cluster are up.

   For example,

   ```
   # gluster nfs-ganesha enable
   Enabling NFS-Ganesha requires Gluster-NFS to be disabled across the
   trusted pool. Do you still want to continue?
    (y/n) y
   This will take a few minutes to complete. Please wait ..
   nfs-ganesha : success
   ```

Note

After enabling NFS-Ganesha, if **`rpcinfo -p`** shows the statd port different from 662, then, restart the statd service:

For Red Hat Enterprise Linux 7:

```
# systemctl restart rpc-statd
```

Tearing down the HA cluster

To tear down the HA cluster, execute the following command:

```
# gluster nfs-ganesha disable
```

For example,

```
# gluster nfs-ganesha disable
Disabling NFS-Ganesha will tear down entire ganesha cluster across the
trusted pool. Do you still want to continue?
(y/n) y
This will take a few minutes to complete. Please wait ..
nfs-ganesha : success
```

Verifying the status of the HA cluster

To verify the status of the HA cluster, execute the following script:

```
# /usr/libexec/ganesha/ganesha-ha.sh --status
/var/run/gluster/shared_storage/nfs-ganesha
```

For example:

```
# /usr/libexec/ganesha/ganesha-ha.sh --status
/var/run/gluster/shared_storage/nfs-ganesha
```

```
 Online: [ server1 server2 server3 server4 ]
server1-cluster_ip-1 server1
server2-cluster_ip-1 server2
server3-cluster_ip-1 server3
server4-cluster_ip-1 server4
Cluster HA Status: HEALTHY
```

Note

- It is recommended to manually restart the **ganesha.nfsd** service after the node is rebooted, to fail back the VIPs.
- Disabling NFS Ganesha does not enable Gluster NFS by default. If required, Gluster NFS must be enabled manually.

6.2.3.2.6. Exporting and Unexporting Volumes through NFS-Ganesha

Exporting Volumes through NFS-Ganesha

To export a Red Hat Gluster Storage volume, execute the following command:

```
# gluster volume set <volname> ganesha.enable on
```

For example:

```
# gluster vol set testvol ganesha.enable on
volume set: success
```

Unexporting Volumes through NFS-Ganesha

To unexport a Red Hat Gluster Storage volume, execute the following command:

```
# gluster volume set <volname> ganesha.enable off
```

This command unexports the Red Hat Gluster Storage volume without affecting other exports.

For example:

```
# gluster vol set testvol ganesha.enable off
volume set: success
```

6.2.3.2.7. Verifying the NFS-Ganesha Status

To verify the status of the volume set options, follow the guidelines mentioned below:

- Check if NFS-Ganesha is started by executing the following commands:

  On Red Hat Enterprise Linux-7

  ```
  # systemctl status nfs-ganesha
  ```

  For example:

  ```
  # systemctl  status nfs-ganesha
     nfs-ganesha.service - NFS-Ganesha file server
     Loaded: loaded (/usr/lib/systemd/system/nfs-ganesha.service; disabled)
     Active: active (running) since Tue 2015-07-21 05:08:22 IST; 19h ago
     Docs: http://github.com/nfs-ganesha/nfs-ganesha/wiki
     Main PID: 15440 (ganesha.nfsd)
     CGroup: /system.slice/nfs-ganesha.service
  ```

```
               └─15440 /usr/bin/ganesha.nfsd -L
/var/log/ganesha/ganesha.log -f /etc/ganesha/ganesha.conf -N NIV_EVENT
   Jul 21 05:08:22 server1 systemd[1]: Started NFS-Ganesha file server.]
```

- Check if the volume is exported.

```
# showmount -e localhost
```

  For example:

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

- The logs of ganesha.nfsd daemon are written to /var/log/ganesha/ganesha.log. Check the log file on noticing any unexpected behavior.

6.2.3.3. Accessing NFS-Ganesha Exports

NFS-Ganesha exports can be accessed by mounting them in either NFSv3 or NFSv4 mode. Since this is an active-active HA configuration, the mount operation can be performed from the VIP of any node.

For better large file performance on all workloads that is generated on Red Hat Enterprise Linux 7 clients, it is recommended to set the following tunable before mounting the volume:

1. Execute the following commands to set the tunable:

```
# sysctl -w sunrpc.tcp_slot_table_entries=128
# echo 128 > /proc/sys/sunrpc/tcp_slot_table_entries
# echo 128 > /proc/sys/sunrpc/tcp_max_slot_table_entries
```

2. To make the tunable persistent on reboot, execute the following commands:

```
# echo "options sunrpc tcp_slot_table_entries=128" >>
/etc/modprobe.d/sunrpc.conf
# echo "options sunrpc tcp_max_slot_table_entries=128" >>
/etc/modprobe.d/sunrpc.conf
```

Note

Ensure that NFS clients and NFS-Ganesha servers in the cluster are DNS resolvable with unique host-names to use file locking through Network Lock Manager (NLM) protocol.

6.2.3.3.1. Mounting exports in NFSv3 Mode

To mount an export in NFSv3 mode, execute the following command:

```
# mount -t nfs -o vers=3 virtual_ip:/volname /mountpoint
```

For example:

```
mount -t nfs -o vers=3 10.70.0.0:/testvol /mnt
```

6.2.3.3.2. Mounting exports in NFSv4 Mode

To mount an export in NFSv4 mode, execute the following command:

```
# mount -t nfs -o vers=4.0 virtual_ip:/volname /mountpoint
```

For example:

```
# mount -t nfs -o vers=4.0 10.70.0.0:/testvol /mnt
```

6.2.3.3.3. Finding clients of an NFS server using dbus

To display the IP addresses of clients that have mounted the NFS exports, execute the following command:

```
# dbus-send --type=method_call --print-reply --system --
dest=org.ganesha.nfsd /org/ganesha/nfsd/ClientMgr
org.ganesha.nfsd.clientmgr.ShowClients
```

Note

If the NFS export is unmounted or if a client is disconnected from the server, it may take a few minutes for this to be updated in the command output.

6.2.3.4. Modifying the NFS-Ganesha HA Setup

To modify the existing HA cluster and to change the default values of the exports use the ganesha-ha.sh script located at /usr/libexec/ganesha/.

6.2.3.4.1. Adding a Node to the Cluster

Before adding a node to the cluster, ensure that the firewall services are enabled as mentioned in *Port Information for NFS-Ganesha* and also the prerequisites mentioned in section *Pre-requisites to run NFS-Ganesha* are met.

Note

Since shared storage and **`/var/lib/glusterd/nfs/secret.pem`** SSH key are already generated, those steps should not be repeated.

To add a node to the cluster, execute the following command on any of the nodes in the existing NFS-Ganesha cluster:

```
# /usr/libexec/ganesha/ganesha-ha.sh --add <HA_CONF_DIR> <HOSTNAME> <NODE-
VIP>
```

where,

HA_CONF_DIR: The directory path containing the ganesha-ha.conf file. By default it is **`/run/gluster/shared_storage/nfs-ganesha.`**

HOSTNAME: Hostname of the new node to be added

NODE-VIP: Virtual IP of the new node to be added.

For example:

```
# /usr/libexec/ganesha/ganesha-ha.sh --add
/var/run/gluster/shared_storage/nfs-ganesha server16 10.00.00.01
```

6.2.3.4.2. Deleting a Node in the Cluster

To delete a node from the cluster, execute the following command on any of the nodes in the existing NFS-Ganesha cluster:

```
# /usr/libexec/ganesha/ganesha-ha.sh --delete <HA_CONF_DIR> <HOSTNAME>
```

where,

HA_CONF_DIR: The directory path containing the ganesha-ha.conf file. By default it is located at **`/run/gluster/shared_storage/nfs-ganesha`**.

HOSTNAME: Hostname of the new node to be added

For example:

```
# /usr/libexec/ganesha/ganesha-ha.sh --delete
/var/run/gluster/shared_storage/nfs-ganesha  server16
```

6.2.3.5. Modifying the Default Export Configurations

It is recommended to use gluster CLI options to export or unexport volumes through NFS-Ganesha. However, this section provides some information on changing configurable parameters in NFS-Ganesha. Such parameter changes require NFS-Ganesha to be started manually.

To modify the default export configurations perform the following steps on any of the nodes in the existing ganesha cluster:

1. Edit/add the required fields in the corresponding export file located at **`/run/gluster/shared_storage/nfs-ganesha/exports/`**.

2. Execute the following command

   ```
   # /usr/libexec/ganesha/ganesha-ha.sh --refresh-config <HA_CONF_DIR>
   <volname>
   ```

where:

- HA_CONF_DIR: The directory path containing the ganesha-ha.conf file. By default it is located at **`/run/gluster/shared_storage/nfs-ganesha`**.

- volname: The name of the volume whose export configuration has to be changed.

Sample export configuration file:

The following are the default set of parameters required to export any entry. The values given here are the default values used by the CLI options to start or stop NFS-Ganesha.

```
# cat export.conf

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

    FSAL {
        name = GLUSTER;
        hostname = "10.xx.xx.xx";  # IP of one of the nodes in the trusted
pool
        volume = "volume_name";     # 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 = "pseudo_path";     # 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
}
```

- Providing Permissions for Specific Clients
- Enabling and Disabling NFSv4 ACLs
- Providing Pseudo Path for NFSv4 Mount
- Exporting Subdirectories

6.2.3.5.1. Providing Permissions for Specific Clients

The parameter values and permission values given in the **EXPORT** block applies to any client that mounts the exported volume. To provide specific permissions to specific clients , introduce a **client** block inside the **EXPORT** block.

For example, to assign specific permissions for client 10.00.00.01, add the following block in the **EXPORT** block.

```
client {
        clients = 10.00.00.01;  # IP of the client.
        access_type = "RO"; # Read-only permissions
        Protocols = "3"; # Allow only NFSv3 protocol.
        anonymous_uid = 1440;
        anonymous_gid = 72;
  }
```

The following section describes various configurations possible via NFS-Ganesha. Minor changes have to be made to the **export.conf** file to see the expected behavior.

All the other clients inherit the permissions that are declared outside the **client** block.

6.2.3.5.2. Enabling and Disabling NFSv4 ACLs

To enable NFSv4 ACLs , edit the following parameter:

```
Disable_ACL = FALSE;
```

Note

NFS clients should remount their share after enabling/disabling ACLs on the NFS-Ganesha server.

6.2.3.5.3. Providing Pseudo Path for NFSv4 Mount

To set NFSv4 pseudo path , edit the below parameter:

```
Pseudo = "pseudo_path"; # NFSv4 pseudo path for this export. Eg:
"/test_volume_pseudo"
```

This path has to be used while mounting the export entry in NFSv4 mode.

6.2.3.5.4. Exporting Subdirectories

Execute the following commands to export subdirectories:

1. Stop the volume by executing the following command:

   ```
   # gluster volume stop <volname>
   ```

2. To export subdirectories within a volume, edit the following parameters in the `export.conf` file.

   ```
   Path = "path_to_subdirectory";  # Path of the volume to be exported.
   Eg: "/test_volume/test_subdir"

    FSAL {
     name = GLUSTER;
     hostname = "10.xx.xx.xx";  # IP of one of the nodes in the trusted
   pool
     volume = "volume_name";  # Volume name. Eg: "test_volume"
     volpath = "path_to_subdirectory_with_respect_to_volume";
   #Subdirectory path from the root of the volume. Eg: "/test_subdir"
    }
   ```

3. Change **`Export_ID`** to an unused value. I should preferably be a larger value so that it cannot be re-used for other volumes.

4. Restart the volume to export the subdirectory.

   ```
   # gluster volume start <volname>
   ```

Note

If there are multiple sub-directories to be exported, create EXPORT blocks for each such sub-directory and then restart the nfs-ganesha service.

6.2.3.5.5. Configuring Upcall Poll Interval

Currently, the default interval (in microseconds) between upcall polls is 10 microseconds. For large numbers of threads, this results in lot of CPU consumption.

To avoid high CPU consumption, you can configure the time interval between two upcall polls by using the **up_poll_usec** option. You can set an optimal value depending on the workload. The default value of **up_poll_usec** option is set to 10 miscroseconds and can be increased upto 60000000 microseconds (60s).

You need to add this option to the FSAL Gluster block in the **export.<volume name>.conf** file.

For example, to set the upcall poll interval to 1 millisecond (1000 microseconds), add the **up_poll_usec** in the FSAL Gluster block in **export.<volume name>.conf** file.

```
FSAL {
      name = GLUSTER;
     hostname = “10.xx.xx.xx.”;
     Volume = "volume_name";
      up_poll_usec = 1000; # Upcall poll interval in microseconds
      }
```

6.2.3.6. Configuring Kerberized NFS-Ganesha

Execute the following steps on all the machines:

1. Install the krb5-workstation and the ntpdate packages on all the machines:

   ```
   # yum install krb5-workstation
   # yum install ntpdate
   ```

   Note

   - The krb5-libs package will be updated as a dependent package.

2. Configure the ntpdate based on the valid time server according to the environment:

   ```
   # echo <valid_time_server> >> /etc/ntp/step-tickers

   # systemctl enable ntpdate

   # systemctl start ntpdate
   ```

3. Ensure that all systems can resolve each other by FQDN in DNS.

4. Configure the **/etc/krb5.conf** file and add relevant changes accordingly. For example:

```
[logging]
  default = FILE:/var/log/krb5libs.log
  kdc = FILE:/var/log/krb5kdc.log
  admin_server = FILE:/var/log/kadmind.log

  [libdefaults]
  dns_lookup_realm = false
  ticket_lifetime = 24h
  renew_lifetime = 7d
  forwardable = true
  rdns = false
  default_realm = EXAMPLE.COM
  default_ccache_name = KEYRING:persistent:%{uid}

  [realms]
  EXAMPLE.COM = {
  kdc = kerberos.example.com
    admin_server = kerberos.example.com
  }

  [domain_realm]
  .example.com = EXAMPLE.COM
   example.com = EXAMPLE.COM
```

Note

For further details regarding the file configuration, refer to **`man krb5.conf`**.

5. On the NFS-server and client, update the /etc/idmapd.conf file by making the required change. For example:

```
Domain = example.com
```

6.2.3.6.1. Setting up the NFS-Ganesha Server:

Execute the following steps to set up the NFS-Ganesha server:

Note

Before setting up the NFS-Ganesha server, make sure to set up the KDC based on the requirements.

1. Install the following packages:

```
# yum install nfs-utils
# yum install rpcbind
```

2. Install the relevant gluster and NFS-Ganesha rpms. For more information see, *Red Hat Gluster Storage 3.4 Installation Guide*.

3. Create a Kerberos principle and add it to krb5.keytab on the NFS-Ganesha server

```
$ kadmin
$ kadmin: addprinc -randkey nfs/<host_name>@EXAMPLE.COM
$ kadmin: ktadd nfs/<host_name>@EXAMPLE.COM
```

For example:

```
# kadmin
Authenticating as principal root/admin@EXAMPLE.COM with password.
Password for root/admin@EXAMPLE.COM:

kadmin:  addprinc -randkey nfs/<host_name>@EXAMPLE.COM
WARNING: no policy specified for nfs/<host_name>@EXAMPLE.COM;
defaulting to no policy
Principal "nfs/<host_name>@EXAMPLE.COM" created.


kadmin:  ktadd nfs/<host_name>@EXAMPLE.COM
Entry for principal nfs/<host_name>@EXAMPLE.COM with kvno2, encryption
type aes256-cts-hmac-sha1-96 added to keytab FILE:/etc/krb5.keytab.
Entry for principal nfs/<host_name>@EXAMPLE.COM with kvno 2,
encryption type aes128-cts-hmac-sha1-96 added to keytab
FILE:/etc/krb5.keytab.
Entry for principal nfs/<host_name>@EXAMPLE.COM with kvno 2,
encryption type des3-cbc-sha1 added to keytab FILE:/etc/krb5.keytab.
Entry for principal nfs/<host_name>@EXAMPLE.COM with kvno 2,
encryption type arcfour-hmac added to keytab FILE:/etc/krb5.keytab.
Entry for principal nfs/<host_name>@EXAMPLE.COM with kvno 2,
encryption type camellia256-cts-cmac added to keytab
FILE:/etc/krb5.keytab.
Entry for principal nfs/<host_name>@EXAMPLE.COM with kvno 2,
encryption type camellia128-cts-cmac added to keytab
FILE:/etc/krb5.keytab.
Entry for principal nfs/<host_name>@EXAMPLE.COM with kvno 2,
encryption type des-hmac-sha1 added to keytab FILE:/etc/krb5.keytab.
Entry for principal nfs/<host_name>@EXAMPLE.COM with kvno 2,
encryption type des-cbc-md5 added to keytab FILE:/etc/krb5.keytab.
```

4. Update **`/etc/ganesha/ganesha.conf`** file as mentioned below:

```
NFS_KRB5
{
        PrincipalName = nfs ;
        KeytabPath = /etc/krb5.keytab ;
        Active_krb5 = true ;
}
```

5. Based on the different kerberos security flavours (krb5, krb5i and krb5p) supported by nfs-ganesha, configure the 'SecType' parameter in the volume export file (/var/run/gluster/shared_storage/nfs-ganesha/exports) with appropriate security flavour

6. Create an unprivileged user and ensure that the users that are created are resolvable to the UIDs through the central user database. For example:

```
# useradd guest
```

Note

The username of this user has to be the same as the one on the NFS-client.

6.2.3.6.2. Setting up the NFS Client

Execute the following steps to set up the NFS client:

Note

For a detailed information on setting up NFS-clients for security on Red Hat Enterprise Linux, see *Section 8.8.2* NFS Security, in the *Red Hat Enterprise Linux 7 Storage Administration Guide*.

1. Install the following packages:

```
# yum install nfs-utils
# yum install rpcbind
```

2. Create a kerberos principle and add it to krb5.keytab on the client side. For example:

```
# kadmin
# kadmin: addprinc -randkey host/<host_name>@EXAMPLE.COM
# kadmin: ktadd host/<host_name>@EXAMPLE.COM
```

```
# kadmin
Authenticating as principal root/admin@EXAMPLE.COM with password.
Password for root/admin@EXAMPLE.COM:

kadmin:  addprinc -randkey host/<host_name>@EXAMPLE.COM
WARNING: no policy specified for host/<host_name>@EXAMPLE.COM;
defaulting to no policy
Principal "host/<host_name>@EXAMPLE.COM" created.

kadmin:  ktadd host/<host_name>@EXAMPLE.COM
Entry for principal host/<host_name>@EXAMPLE.COM with kvno 2,
encryption type aes256-cts-hmac-sha1-96 added to keytab
FILE:/etc/krb5.keytab.
Entry for principal host/<host_name>@EXAMPLE.COM with kvno 2,
encryption type aes128-cts-hmac-sha1-96 added to keytab
FILE:/etc/krb5.keytab.
Entry for principal host/<host_name>@EXAMPLE.COM with kvno 2,
encryption type des3-cbc-sha1 added to keytab FILE:/etc/krb5.keytab.
Entry for principal host/<host_name>@EXAMPLE.COM with kvno 2,
encryption type arcfour-hmac added to keytab FILE:/etc/krb5.keytab.
Entry for principal host/<host_name>@EXAMPLE.COM with kvno 2,
encryption type camellia256-cts-cmac added to keytab
FILE:/etc/krb5.keytab.
Entry for principal host/<host_name>@EXAMPLE.COM with kvno 2,
encryption type camellia128-cts-cmac added to keytab
FILE:/etc/krb5.keytab.
```

```
Entry for principal host/<host_name>@EXAMPLE.COM with kvno 2,
encryption type des-hmac-sha1 added to keytab FILE:/etc/krb5.keytab.
Entry for principal host/<host_name>@EXAMPLE.COM with kvno 2,
encryption type des-cbc-md5 added to keytab FILE:/etc/krb5.keytab.
```

3. Check the status of nfs-client.target service and start it, if not already started:

   ```
   # systemctl status nfs-client.target
   # systemctl start nfs-client.target
   # systemctl enable nfs-client.target
   ```

4. Create an unprivileged user and ensure that the users that are created are resolvable to the UIDs through the central user database. For example:

   ```
   # useradd guest
   ```

   Note

   The username of this user has to be the same as the one on the NFS-server.

5. Mount the volume specifying kerberos security type:

   ```
   # mount -t nfs -o sec=krb5 <host_name>:/testvolume /mnt
   ```

   As root, all access should be granted.

   For example:

   Creation of a directory on the mount point and all other operations as root should be successful.

   ```
   # mkdir <directory name>
   ```

6. Login as a guest user:

   ```
   # su - guest
   ```

   Without a kerberos ticket, all access to /mnt should be denied. For example:

   ```
   # su guest
   # ls
   ls: cannot open directory .: Permission denied
   ```

7. Get the kerberos ticket for the guest and access /mnt:

   ```
   # kinit
   Password for guest@EXAMPLE.COM:

   # ls
   <directory created>
   ```

Important

With this ticket, some access must be allowed to /mnt. If there are directories on the NFS-server where "guest" does not have access to, it should work correctly.

6.2.3.7. NFS-Ganesha Service Downtime

In a highly available active-active environment, if a NFS-Ganesha server that is connected to a NFS client running a particular application goes down, the application/NFS client is seamlessly connected to another NFS-Ganesha server without any administrative intervention. However, there is a delay or fail-over time in connecting to another NFS-Ganesha server. This delay can be experienced during fail-back too, that is, when the connection is reset to the original node/server.

The following list describes how the time taken for the NFS server to detect a server reboot or resume is calculated.

- If the ganesha.nfsd dies (crashes, oomkill, admin kill), the maximum time to detect it and put the ganesha cluster into grace is 20sec, plus whatever time pacemaker needs to effect the fail-over.

  Note

  This time taken to detect if the service is down, can be edited using the following command on all the nodes:

  ```
  # pcs resource op remove nfs-mon monitor
  # pcs resource op add nfs-mon monitor interval=
  <interval_period_value>
  ```

- If the whole node dies (including network failure) then this down time is the total of whatever time pacemaker needs to detect that the node is gone, the time to put the cluster into grace, and the time to effect the fail-over. This is ~20 seconds.

- So the max-fail-over time is approximately 20-22 seconds, and the average time is typically less. In other words, the time taken for NFS clients to detect server reboot or resume I/O is 20 - 22 seconds.

6.2.3.7.1. Modifying the Fail-over Time

After failover, there is a short period of time during which clients try to reclaim their lost OPEN/LOCK state. Servers block certain file operations during this period, as per the NFS specification. The file operations blocked are as follows:

Table 6.6.

Protocols	FOPs
NFSV3	- SETATTR

NLM	LOCK UNLOCK SHARE UNSHARE CANCEL LOCKT
NFSV4	LOCK LOCKT OPEN REMOVE RENAME SETATTR

> **Note**
>
> LOCK, SHARE, and UNSHARE will be blocked only if it is requested with reclaim set to FALSE.
>
> OPEN will be blocked if requested with claim type other than CLAIM_PREVIOUS or CLAIM_DELEGATE_PREV.

The default value for the grace period is 90 seconds. This value can be changed by adding the following lines in the **/etc/ganesha/ganesha.conf** file.

```
NFSv4 {
Grace_Period=<grace_period_value_in_sec>;
}
```

After editing the **/etc/ganesha/ganesha.conf** file, restart the NFS-Ganesha service using the following command on all the nodes :

On Red Hat Enterprise Linux 7

```
# systemctl restart nfs-ganesha
```

6.2.3.8. Manually Configuring NFS-Ganesha Exports

It is recommended to use gluster CLI options to export or unexport volumes through NFS-Ganesha. However, this section provides some information on changing configurable parameters in NFS-Ganesha. Such parameter changes require NFS-Ganesha to be started manually.

To modify the default export configurations perform the following steps on any of the nodes in the existing ganesha cluster:

1. Edit/add the required fields in the corresponding export configuration file in the **/run/gluster/shared_storage/nfs-ganesha/exports** directory.

2. Execute the following command

```
# /usr/libexec/ganesha/ganesha-ha.sh --refresh-config <HA_CONF_DIR>
<volname>
```

where:

- HA_CONF_DIR: The directory path containing the ganesha-ha.conf file. By default it is located at **`/etc/ganesha`**.
- volname: The name of the volume whose export configuration has to be changed.

Sample export configuration file:

The following are the default set of parameters required to export any entry. The values given here are the default values used by the CLI options to start or stop NFS-Ganesha.

```
# cat export.conf

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

    FSAL {
        name = GLUSTER;
        hostname = "10.xx.xx.xx";  # IP of one of the nodes in the trusted
pool
        volume = "volume_name";     # 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 = "pseudo_path";     # 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
}
```

The following section describes various configurations possible via NFS-Ganesha. Minor changes have to be made to the **`export.conf`** file to see the expected behavior.

- Exporting Subdirectories
- Providing Permissions for Specific Clients
- Enabling and Disabling NFSv4 ACLs
- Providing Pseudo Path for NFSv4 Mount

Exporting Subdirectories

To export subdirectories within a volume, edit the following parameters in the **`export.conf`** file.

```
Path = "path_to_subdirectory";  # Path of the volume to be exported. Eg:
"/test_volume/test_subdir"
```

```
 FSAL {
  name = GLUSTER;
  hostname = "10.xx.xx.xx";  # IP of one of the nodes in the trusted pool
  volume = "volume_name";  # Volume name. Eg: "test_volume"
  volpath = "path_to_subdirectory_with_respect_to_volume"; #Subdirectory
path from the root of the volume. Eg: "/test_subdir"
 }
```

Providing Permissions for Specific Clients

The parameter values and permission values given in the **`EXPORT`** block applies to any client that mounts the exported volume. To provide specific permissions to specific clients , introduce a **`client`** block inside the **`EXPORT`** block.

For example, to assign specific permissions for client 10.00.00.01, add the following block in the **`EXPORT`** block.

```
client {
        clients = 10.00.00.01;  # IP of the client.
        allow_root_access = true;
        access_type = "RO"; # Read-only permissions
        Protocols = "3"; # Allow only NFSv3 protocol.
        anonymous_uid = 1440;
        anonymous_gid = 72;
  }
```

All the other clients inherit the permissions that are declared outside the **`client`** block.

Enabling and Disabling NFSv4 ACLs

To enable NFSv4 ACLs , edit the following parameter:

```
Disable_ACL = FALSE;
```

Providing Pseudo Path for NFSv4 Mount

To set NFSv4 pseudo path , edit the below parameter:

```
Pseudo = "pseudo_path"; # NFSv4 pseudo path for this export. Eg:
"/test_volume_pseudo"
```

This path has to be used while mounting the export entry in NFSv4 mode.

6.2.3.9. Troubleshooting

Mandatory checks

Ensure you execute the following commands for all the issues/failures that is encountered:

- Make sure all the prerequisites are met.
- Execute the following commands to check the status of the services:

```
# service nfs-ganesha status
# service pcsd status
# service pacemaker status
# pcs status
```

- Review the followings logs to understand the cause of failure.

```
/var/log/ganesha/ganesha.log
/var/log/ganesha/ganesha-gfapi.log
/var/log/messages
/var/log/pcsd.log
```

- **Situation**

  NFS-Ganesha fails to start.

  Solution

  Ensure you execute all the mandatory checks to understand the root cause before proceeding with the following steps. Follow the listed steps to fix the issue:

  - Ensure the kernel and gluster nfs services are inactive.

  - Ensure that the port 875 is free to connect to the RQUOTA service.

  - Ensure that the shared storage volume mount exists on the server after node reboot/shutdown. If it does not, then mount the shared storage volume manually using the following command:

    ```
    # mount -t glusterfs <local_node's_hostname>:gluster_shared_storage
    /var/run/gluster/shared_storage
    ```

  For more information see, section *Manually Configuring NFS-Ganesha Exports.*

- **Situation**

  NFS-Ganesha port 875 is unavailable.

  Solution

  Ensure you execute all the mandatory checks to understand the root cause before proceeding with the following steps. Follow the listed steps to fix the issue:

  - Run the following command to extract the PID of the process using port 875:

    ```
    netstat -anlp | grep 875
    ```

  - Determine if the process using port 875 is an important system or user process.

  - Perform one of the following depending upon the importance of the process:

    - If the process using port 875 is an important system or user process:

      - Assign a different port to this service by modifying following line in '/etc/ganesha/ganesha.conf' file on all the nodes:

```
        # Use a non-privileged port for RQuota
        Rquota_Port = port_number;
        ```

    - Run the following commands after modifying the port number:

        ```
        # semanage port -a -t mountd_port_t -p tcp port_number
        # semanage port -a -t mountd_port_t -p udp port_number
        ```

    - Run the following command to restart NFS-Ganesha:

        ```
        systemctl restart nfs-ganesha
        ```

- If the process using port 875 is not an important system or user process:

    - Run the following command to kill the process using port 875:

        ```
        # kill pid;
        ```

        Use the process ID extracted from the previous step.

    - Run the following command to ensure that the process is killed and port 875 is free to use:

        ```
        # ps aux | grep pid;
        ```

    - Run the following command to restart NFS-Ganesha:

        ```
        systemctl restart nfs-ganesha
        ```

    - If required, restart the killed process.

- **Situation**

    NFS-Ganesha Cluster setup fails.

    Solution

    Ensure you execute all the mandatory checks to understand the root cause before proceeding with the following steps.

    - Ensure the kernel and gluster nfs services are inactive.

    - Ensure that **`pcs cluster auth`** command is executed on all the nodes with same password for the user **`hacluster`**

    - Ensure that shared volume storage is mounted on all the nodes.

    - Ensure that the name of the HA Cluster does not exceed 15 characters.

    - Ensure UDP multicast packets are pingable using **`OMPING`**.

    - Ensure that Virtual IPs are not assigned to any NIC.

- **Situation**

NFS-Ganesha has started and fails to export a volume.

Solution

Ensure you execute all the mandatory checks to understand the root cause before proceeding with the following steps. Follow the listed steps to fix the issue:

- Ensure that volume is in **`Started`** state using the following command:

  ```
  # gluster volume status <volname>
  ```

- Execute the following commands to check the status of the services:

  ```
  # service nfs-ganesha status
  # showmount -e localhost
  ```

- Review the followings logs to understand the cause of failure.

  ```
  /var/log/ganesha/ganesha.log
  /var/log/ganesha/ganesha-gfapi.log
  /var/log/messages
  ```

- Ensure that dbus service is running using the following command

  ```
  # service messagebus status
  ```

- If the volume is not in a started state, run the following command to start the volume.

  ```
  # gluster volume start <volname>
  ```

  If the volume is not exported as part of volume start, run the following command to re-export the volume:

  ```
  # /usr/libexec/ganesha/dbus-send.sh /var/run/gluster/shared_storage
  on <volname>
  ```

- ### Situation

  Adding a new node to the HA cluster fails.

  ### Solution

  Ensure you execute all the mandatory checks to understand the root cause before proceeding with the following steps. Follow the listed steps to fix the issue:

  - Ensure to run the following command from one of the nodes that is already part of the cluster:

    ```
    # ganesha-ha.sh --add <HA_CONF_DIR>  <NODE-HOSTNAME>  <NODE-VIP>
    ```

  - Ensure that gluster_shared_storage volume is mounted on the node that needs to be added.

  - Make sure that all the nodes of the cluster is DNS resolvable from the node that needs to be added.

- Execute the following command for each of the hosts in the HA cluster on the node that needs to be added:

  ```
  # pcs cluster auth <hostname>
  ```

- **Situation**

  Cleanup required when nfs-ganesha HA cluster setup fails.

  Solution

  To restore back the machines to the original state, execute the following commands on each node forming the cluster:

  ```
  # /usr/libexec/ganesha/ganesha-ha.sh --teardown
  /var/run/gluster/shared_storage/nfs-ganesha
  # /usr/libexec/ganesha/ganesha-ha.sh --cleanup
  /var/run/gluster/shared_storage/nfs-ganesha
  # systemctl stop nfs-ganesha
  ```

- **Situation**

  Permission issues.

  Solution

  By default, the **`root squash`** option is disabled when you start NFS-Ganesha using the CLI. In case, you encounter any permission issues, check the unix permissions of the exported entry.

6.3. SMB

You can access Red Hat Gluster Storage volumes using the Server Message Block (SMB) protocol by exporting directories in Red Hat Gluster Storage volumes as SMB shares on the server.

This section describes how to enable SMB shares, how to mount SMB shares manually and automatically on Microsoft Windows and macOS based clients, and how to verify that the share has been mounted successfully.

> **Important**
>
> To export Red Hat Gluster Storage volumes with Samba, setting up CTDB is mandatory.

Follow the process outlined in Overview of configuring SMB shares. The details of this overview are provided in the rest of this section.

Overview of configuring SMB shares

1. Verify that your system fulfils the requirements outlined in Section 6.3.1, “Requirements for using SMB with Red Hat Gluster Storage”.

2. Configure CTDB to export Red Hat Gluster Storage volumes with Samba: Section 6.3.2, “Setting up CTDB for Samba”.

3. Configure your volumes to be shared using SMB: Section 6.3.3, “Sharing Volumes over SMB”.

4. If you want to mount volumes on macOS clients: Section 6.3.4.1, “Configuring the Apple Create Context for macOS users”.

5. Set up permissions for user access: Section 6.3.4.2, “Configuring read/write access for a non-privileged user”.

6. Mount the shared volume on a client:

   - Section 6.3.5.1, “Manually mounting volumes exported with SMB on Red Hat Enterprise Linux”
   - Section 6.3.5.4, “Configuring automatic mounting for volumes exported with SMB on Red Hat Enterprise Linux”
   - Section 6.3.5.2.1, “Using Microsoft Windows Explorer to manually mount a volume”
   - Section 6.3.5.2.2, “Using Microsoft Windows command line interface to manually mount a volume”
   - Section 6.3.5.5, “Configuring automatic mounting for volumes exported with SMB on Microsoft Windows”
   - Section 6.3.5.3, “Manually mounting volumes exported with SMB on macOS”
   - Section 6.3.5.6, “Configuring automatic mounting for volumes exported with SMB on macOS”

7. Verify that your shared volume is working properly: Section 6.3.6, “Starting and Verifying your Configuration”

6.3.1. Requirements for using SMB with Red Hat Gluster Storage

- Samba is the server software used to export Linux filesystems with the SMB protocol. For exporting Red Hat Gluster Storage volumes with Samba, it is mandatory to have CTDB configured, which is a component of Samba. For information on subscribing to the correct channels for SMB support, see Subscribing to the Red Hat Gluster Storage server channels in the *Red Hat Gluster Storage 3.4 Installation Guide.*

- Enable the Samba firewall service in the active zones for runtime and permanent mode. The following commands are for systems based on Red Hat Enterprise Linux 7.

  To get a list of active zones, run the following command:

  ```
  # firewall-cmd --get-active-zones
  ```

  To allow the firewall services in the active zones, run the following commands

  ```
  # firewall-cmd --zone=zone_name --add-service=samba
  # firewall-cmd --zone=zone_name --add-service=samba  --permanent
  ```

6.3.2. Setting up CTDB for Samba

To export Red Hat Gluster Storage volumes with Samba, configure CTDB (Cluster Trivial Database).

CTDB provides high availability by adding virtual IP addresses (VIPs) and a heartbeat service. When a node in the trusted storage pool fails, CTDB enables a different node to take over the virtual IP addresses that the failed node was hosting. This ensures the IP addresses for the services provided are always available.

Important

Amazon Elastic Compute Cloud (EC2) does not support VIPs and is hence not compatible with this solution.

Prerequisites

- If you already have an older version of CTDB (version <= ctdb1.x), then remove CTDB by executing the following command:

  ```
  # yum remove ctdb
  ```

  After removing the older version, proceed with installing the latest CTDB.

  Note

  Ensure that the system is subscribed to the samba channel to get the latest CTDB packages.

- Install CTDB on all the nodes that are used as Samba servers to the latest version using the following command:

  ```
  # yum install ctdb
  ```

- In a CTDB based high availability environment of Samba , the locks will not be migrated on failover.

- Enable the CTDB firewall service in the active zones for runtime and permanent mode. The following commands are for systems based on Red Hat Enterprise Linux 7.

  To get a list of active zones, run the following command:

  ```
  # firewall-cmd --get-active-zones
  ```

  To add ports to the active zones, run the following commands:

  ```
  # firewall-cmd --zone=zone_name --add-port=4379/tcp
  # firewall-cmd --zone=zone_name --add-port=4379/tcp  --permanent
  ```

Configuring CTDB on Red Hat Gluster Storage Server

1. Create a new replicated volume to house the CTDB lock file. The lock file has a size of zero bytes, so use small bricks.

   To create a replicated volume run the following command, replacing *N* with the number of nodes to replicate across:

   ```
   # gluster volume create volname replica N ip_address_1:brick_path ...
   ip_address_N:brick_path
   ```

   For example:

```
# gluster volume create ctdb replica 3
10.16.157.75:/rhgs/brick1/ctdb/b1 10.16.157.78:/rhgs/brick1/ctdb/b2
10.16.157.81:/rhgs/brick1/ctdb/b3
```

2. In the following files, replace **`all`** in the statement **`META="all"`** with the newly created volume name, for example, **`META="ctdb"`**.

   ```
   /var/lib/glusterd/hooks/1/start/post/S29CTDBsetup.sh
   /var/lib/glusterd/hooks/1/stop/pre/S29CTDB-teardown.sh
   ```

3. In the **`/etc/samba/smb.conf`** file, add the following line in the global section on all the nodes:

   ```
   clustering=yes
   ```

4. Start the volume.

   The S29CTDBsetup.sh script runs on all Red Hat Gluster Storage servers, adds an entry in **`/etc/fstab`** for the mount, and mounts the volume at **`/gluster/lock`** on all the nodes with Samba server. It also enables automatic start of CTDB service on reboot.

   Note

   When you stop the special CTDB volume, the S29CTDB-teardown.sh script runs on all Red Hat Gluster Storage servers and removes an entry in **`/etc/fstab`** for the mount and unmounts the volume at **`/gluster/lock`**.

5. Verify that the **`/etc/sysconfig/ctdb`** file exists on all nodes that are used as a Samba server. This file contains CTDB configuration details recommended for Red Hat Gluster Storage.

6. Create the **`/etc/ctdb/nodes`** file on all the nodes that are used as Samba servers and add the IP addresses of these nodes to the file.

   ```
   10.16.157.0
   10.16.157.3
   10.16.157.6
   ```

   The IP addresses listed here are the private IP addresses of Samba servers.

7. On nodes that are used as Samba servers and require IP failover, create the **`/etc/ctdb/public_addresses`** file. Add any virtual IP addresses that CTDB should create to the file in the following format:

   ```
   VIP/routing_prefix network_interface
   ```

   For example:

   ```
   192.168.1.20/24 eth0
   192.168.1.21/24 eth0
   ```

8. Start the CTDB service on all the nodes.

   ```
   # service ctdb start
   ```

6.3.3. Sharing Volumes over SMB

After you follow this process, any gluster volumes configured on servers that run Samba are exported automatically on starting the volume.

The procedure to share volumes over samba differs on the Samba version you choose.

If you are using an older version of Samba:

1. Enable SMB specific caching:

   ```
   # gluster volume set VOLNAME performance.cache-samba-metadata on
   ```

   You can also enable generic metadata caching to improve performance. See Section 21.7, “Directory Operations” for details.

2. Restart the **glusterd** service on each Red Hat Gluster Storage node.

3. Verify proper lock and I/O coherence:

   ```
   # gluster volume set VOLNAME storage.batch-fsync-delay-usec 0
   ```

If you are using Samba-4.8.5-104 or later:

1. To export gluster volume as SMB share via Samba, one of the following volume options, **user.cifs** or **user.smb** is required.

   To enable user.cifs volume option, run:

   ```
   # gluster VOLNAME set user.cifs enable
   ```

   And to enable user.smb, run:

   ```
   # gluster VOLNAME set user.smb enable
   ```

   Red Hat Gluster Storage 3.4 introduces a group command **samba** for configuring the necessary volume options for Samba-CTDB setup.

2. Execute the following command to configure the volume options for the Samba-CTDB:

   ```
   # gluster volume set VOLNAME group samba
   ```

   This command will enable the following option for Samba-CTDB setup:

   - performance.readdir-ahead: on
   - performance.parallel-readdir: on
   - performance.nl-cache-timeout: 600
   - performance.nl-cache: on
   - performance.cache-samba-metadata: on
   - network.inode-lru-limit: 200000

- performance.md-cache-timeout: 600
- performance.cache-invalidation: on
- features.cache-invalidation-timeout: 600
- features.cache-invalidation: on
- performance.stat-prefetch: on

Then, for all Samba versions:

1. Verify that the volume can be accessed from the SMB/CIFS share:

   ```
   # smbclient -L <hostname> -U%
   ```

   For example:

   ```
   # smbclient -L rhs-vm1 -U%
   Domain=[MYGROUP] OS=[Unix] Server=[Samba 4.1.17]

        Sharename       Type      Comment
        ---------       ----      -------
        IPC$            IPC       IPC Service (Samba Server Version
   4.1.17)
        gluster-vol1    Disk      For samba share of volume vol1
   Domain=[MYGROUP] OS=[Unix] Server=[Samba 4.1.17]

        Server               Comment
        ---------            -------

        Workgroup            Master
        ---------            -------
   ```

2. Verify that the SMB/CIFS share can be accessed by the user, run the following command:

   ```
   # smbclient //VIP/gluster-volname -U username%password
   ```

   For example:

   ```
   # smbclient //10.0.0.1/gluster-vol1 -U root%redhat
   Domain=[MYGROUP] OS=[Unix] Server=[Samba 4.1.17]
   smb: \> mkdir test
   smb: \> cd test\
   smb: \test\> pwd
   Current directory is \\10.0.0.1\gluster-vol1\test\
   smb: \test\>
   ```

3. Configure this share so that a client can mount it using the address of any server in the trusted storage pool that provides this volume.

   a. Open the **/etc/samba/smb.conf** file in a text editor and add the following lines for a simple configuration:

      ```
      [gluster-VOLNAME]
      ```

```
comment = For samba share of volume VOLNAME
vfs objects = glusterfs
glusterfs:volume = VOLNAME
glusterfs:logfile = /var/log/samba/VOLNAME.log
glusterfs:loglevel = 7
path = /
read only = no
guest ok = yes
```

The configuration options are described in the following table:

Table 6.7. Configuration Options

Configuration Options	Required?	Default Value	Description
Path	Yes	n/a	It represents the path that is relative to the root of the gluster volume that is being shared. Hence **/** represents the root of the gluster volume. Exporting a subdirectory of a volume is supported and /subdir in path exports only that subdirectory of the volume.
glusterfs:volume	Yes	n/a	The volume name that is shared.
glusterfs:logfile	No	NULL	Path to the log file that will be used by the gluster modules that are loaded by the vfs plugin. Standard Samba variable substitutions as mentioned in **smb.conf** are supported.
glusterfs:loglevel	No	7	This option is equivalent to the **client-log-level** option of gluster. 7 is the default value and corresponds to the INFO level.

Configuration Options	Required?	Default Value	Description
glusterfs:volfile_server	No	localhost	The gluster server to be contacted to fetch the **volfile** for the volume. It takes the value, which is a list of white space separated elements, where each element is **unix+/path/to/socket/file** or **[tcp+]IP|hostname|\[IPv6\][:port]**

b. Run **service smb [re]start** to start or restart the **smb** service.

c. Run **smbpasswd** to set the SMB password.

```
# smbpasswd -a username
```

Specify the SMB password. This password is used during the SMB mount.

6.3.4. Configuring User Access to Shared Volumes

6.3.4.1. Configuring the Apple Create Context for macOS users

1. Add the following lines to the **[global]** section of the **smb.conf** file. Note that the indentation level shown is required.

```
              fruit:aapl = yes
              ea support = yes
```

2. Load the **vfs_fruit** module and its dependencies by adding the following line to your volume's export configuration block in the **smb.conf** file.

```
vfs objects = fruit streams_xattr glusterfs
```

For example:

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

fruit:encoding = native
```

6.3.4.2. Configuring read/write access for a non-privileged user

1. Add the user on all the Samba servers based on your configuration:

   ```
   # adduser username
   ```

2. Add the user to the list of Samba users on all Samba servers and assign password by executing the following command:

   ```
   # smbpasswd -a username
   ```

3. From any other Samba server, mount the volume using the FUSE protocol.

   ```
   # mount -t glusterfs -o acl ip-address:/volname /mountpoint
   ```

   For example:

   ```
   # mount -t glusterfs -o acl rhs-a:/repvol /mnt
   ```

4. Use the **`setfacl`** command to provide the required permissions for directory access to the user.

   ```
   # setfacl -m user:username:rwx mountpoint
   ```

   For example:

   ```
   # setfacl -m user:cifsuser:rwx /mnt
   ```

6.3.5. Mounting Volumes using SMB

6.3.5.1. Manually mounting volumes exported with SMB on Red Hat Enterprise Linux

1. Install the **`cifs-utils`** package on the client.

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

2. Run **`mount -t cifs`** to mount the exported SMB share, using the syntax example as guidance.

   ```
   # mount -t cifs -o user=username,pass=password
   //hostname/gluster-volname /mountpoint
   ```

   The **`sec=ntlmssp`** parameter is also required when mounting a volume on Red Hat Enterprise Linux 6.

   ```
   # mount -t cifs -o user=username,pass=password,sec=ntlmssp
   //hostname/gluster-volname /mountpoint
   ```

   For example:

   ```
   # mount -t cifs -o user=cifsuser,pass=redhat,sec=ntlmssp
   //server1/gluster-repvol /cifs
   ```

3. Run `# smbstatus -S` on the server to display the status of the volume:

```
Service        pid     machine             Connected at
------------------------------------------------------------------
-
gluster-VOLNAME 11967   __ffff_192.168.1.60  Mon Aug  6 02:23:25 2012
```

6.3.5.2. Manually mounting volumes exported with SMB on Microsoft Windows

6.3.5.2.1. Using Microsoft Windows Explorer to manually mount a volume

1. In Windows Explorer, click **Tools** → **Map Network Drive…**. to open the **`Map Network Drive`** screen.

2. Choose the drive letter using the **Drive** drop-down list.

3. In the **`Folder`** text box, specify the path of the server and the shared resource in the following format: \\*SERVER_NAME**VOLNAME*.

4. Click **`Finish`** to complete the process, and display the network drive in Windows Explorer.

5. Navigate to the network drive to verify it has mounted correctly.

6.3.5.2.2. Using Microsoft Windows command line interface to manually mount a volume

1. Click **Start** → **Run**, and then type **`cmd`**.

2. Enter **`net use z: \\SERVER_NAME\VOLNAME`**, where *z:* is the drive letter to assign to the shared volume.

   For example, **`net use y: \\server1\test-volume`**

3. Navigate to the network drive to verify it has mounted correctly.

6.3.5.3. Manually mounting volumes exported with SMB on macOS

Prerequisites

- Ensure that your Samba configuration allows the use the SMB Apple Create Context.

- Ensure that the username you're using is on the list of allowed users for the volume.

Manual mounting process

1. In the **Finder**, click **`Go`** > **`Connect to Server`**.

2. In the **`Server Address`** field, type the IP address or hostname of a Red Hat Gluster Storage server that hosts the volume you want to mount.

3. Click **`Connect`**.

4. When prompted, select **`Registered User`** to connect to the volume using a valid username and password.

   If required, enter your user name and password, then select the server volumes or shared folders that you want to mount.

To make it easier to connect to the computer in the future, select **Remember this password in my keychain** to add your user name and password for the computer to your keychain.

For further information about mounting volumes on macOS, see the Apple Support documentation: https://support.apple.com/kb/PH25269?locale=en_US.

6.3.5.4. Configuring automatic mounting for volumes exported with SMB on Red Hat Enterprise Linux

1. Open the **/etc/fstab** file in a text editor and add a line containing the following details:

   ```
   \\HOSTNAME|IPADDRESS\SHARE_NAME MOUNTDIR cifs OPTIONS DUMP FSCK
   ```

   In the *OPTIONS* column, ensure that you specify the **credentials** option, with a value of the path to the file that contains the username and/or password.

   Using the example server names, the entry contains the following replaced values.

   ```
   \\server1\test-volume /mnt/glusterfs cifs
   credentials=/etc/samba/passwd,_netdev 0 0
   ```

   The **sec=ntlmssp** parameter is also required when mounting a volume on Red Hat Enterprise Linux 6, for example:

   ```
   \\server1\test-volume /mnt/glusterfs cifs
   credentials=/etc/samba/passwd,_netdev,sec=ntlmssp 0 0
   ```

   See the **mount.cifs** man page for more information about these options.

2. Run **# smbstatus -S** on the client to display the status of the volume:

   ```
   Service        pid     machine             Connected at
   -------------------------------------------------------------------
   -
   gluster-VOLNAME 11967   __ffff_192.168.1.60  Mon Aug  6 02:23:25 2012
   ```

6.3.5.5. Configuring automatic mounting for volumes exported with SMB on Microsoft Windows

1. In Windows Explorer, click **Tools** → **Map Network Drive…**. to open the **Map Network Drive** screen.

2. Choose the drive letter using the **Drive** drop-down list.

3. In the **Folder** text box, specify the path of the server and the shared resource in the following format: \\*SERVER_NAME**VOLNAME*.

4. Click the **Reconnect at logon** check box.

5. Click **Finish** to complete the process, and display the network drive in Windows Explorer.

6. If the **Windows Security** screen pops up, enter the username and password and click **OK**.

7. Navigate to the network drive to verify it has mounted correctly.

6.3.5.6. Configuring automatic mounting for volumes exported with SMB on macOS

1. Manually mount the volume using the process outlined in Section 6.3.5.3, “Manually mounting volumes exported with SMB on macOS”.

2. In the **Finder**, click **`System Preferences`** > **`Users & Groups`** > **`Username`** > **`Login Items`**.

3. Drag and drop the mounted volume into the login items list.

   Check **`Hide`** if you want to prevent the drive's window from opening every time you boot or log in.

For further information about mounting volumes on macOS, see the Apple Support documentation: https://support.apple.com/kb/PH25269?locale=en_US.

6.3.6. Starting and Verifying your Configuration

Perform the following to start and verify your configuration:

Verify the Configuration

Verify the virtual IP (VIP) addresses of a shut down server are carried over to another server in the replicated volume.

1. Verify that CTDB is running using the following commands:

   ```
   # ctdb status
   # ctdb ip
   # ctdb ping -n all
   ```

2. Mount a Red Hat Gluster Storage volume using any one of the VIPs.

3. Run **`# ctdb ip`** to locate the physical server serving the VIP.

4. Shut down the CTDB VIP server to verify successful configuration.

   When the Red Hat Gluster Storage server serving the VIP is shut down there will be a pause for a few seconds, then I/O will resume.

6.3.7. Disabling SMB Shares

To stop automatic sharing on all nodes for all volumes execute the following steps:

1. On all Red Hat Gluster Storage Servers, with elevated privileges, navigate to /var/lib/glusterd/hooks/1/start/post

2. Rename the S30samba-start.sh to K30samba-start.sh.

   For more information about these scripts, see Section 13.2, “Prepackaged Scripts”.

To stop automatic sharing on all nodes for one particular volume:

1. Run the following command to disable automatic SMB sharing per-volume:

   ```
   # gluster volume set <VOLNAME> user.smb disable
   ```

6.3.8. Accessing Snapshots in Windows

A snapshot is a read-only point-in-time copy of the volume. Windows has an inbuilt mechanism to browse snapshots via Volume Shadow-copy Service (also known as VSS). Using this feature users can access the previous versions of any file or folder with minimal steps.

Note

Shadow Copy (also known as Volume Shadow-copy Service, or VSS) is a technology included in Microsoft Windows that allows taking snapshots of computer files or volumes, apart from viewing snapshots. Currently we only support viewing of snapshots. Creation of snapshots with this interface is NOT supported.

6.3.8.1. Configuring Shadow Copy

To configure shadow copy, the following configurations must be modified/edited in the smb.conf file. The smb.conf file is located at etc/samba/smb.conf.

Note

Ensure, shadow_copy2 module is enabled in smb.conf. To enable add the following parameter to the vfs objects option.

For example:

```
vfs objects = shadow_copy2 glusterfs
```

Table 6.8. Configuration Options

Configuration Options	Required?	Default Value	Description
shadow:snapdir	Yes	n/a	Path to the directory where snapshots are kept. The snapdir name should be .snaps.
shadow:basedir	Yes	n/a	Path to the base directory that snapshots are from. The basedir value should be /.
shadow:sort	Optional	unsorted	The supported values are asc/desc. By this parameter one can specify that the shadow copy directories should be sorted before they are sent to the client. This can be beneficial as unix filesystems are usually not listed alphabetically sorted. If enabled, it is specified in descending order.

Configuration Options	Required?	Default Value	Description
shadow:localtime	Optional	UTC	This is an optional parameter that indicates whether the snapshot names are in UTC/GMT or in local time.
shadow:format	Yes	n/a	This parameter specifies the format specification for the naming of snapshots. The format must be compatible with the conversion specifications recognized by str[fp]time. The default value is _GMT-%Y.%m.%d-%H.%M.%S.
shadow:fixinodes	Optional	No	If you enable shadow:fixinodes then this module will modify the apparent inode number of files in the snapshot directories using a hash of the files path. This is needed for snapshot systems where the snapshots have the same device:inode number as the original files (such as happens with GPFS snapshots). If you don't set this option then the 'restore' button in the shadow copy UI will fail with a sharing violation.
shadow:snapprefix	Optional	n/a	Regular expression to match prefix of snapshot name. Red Hat Gluster Storage only supports Basic Regular Expression (BRE)
shadow:delimiter	Optional	_GMT	delimiter is used to separate shadow:snapprefix and shadow:format.

Following is an example of the smb.conf file:

```
[gluster-vol0]
comment = For samba share of volume vol0
vfs objects = shadow_copy2 glusterfs
glusterfs:volume = vol0
glusterfs:logfile = /var/log/samba/glusterfs-vol0.%M.log
glusterfs:loglevel = 3
```

```
path = /
read only = no
guest ok = yes
shadow:snapdir = /.snaps
shadow:basedir = /
shadow:sort = desc
shadow:snapprefix= ^S[A-Za-z0-9]*p$
shadow:format = _GMT-%Y.%m.%d-%H.%M.%S
```

In the above example, the mentioned parameters have to be added in the smb.conf file to enable shadow copy. The options mentioned are not mandatory.

Shadow copy will filter all the snapshots based on the smb.conf entries. It will only show those snapshots which matches the criteria. In the example mentioned earlier, the snapshot name should start with an 'S' and end with 'p' and any alpha numeric characters in between is considered for the search. For example in the list of the following snapshots, the first two snapshots will be shown by Windows and the last one will be ignored. Hence, these options will help us filter out what snapshots to show and what not to.

```
Snap_GMT-2016.06.06-06.06.06
Sl123p_GMT-2016.07.07-07.07.07
xyz_GMT-2016.08.08-08.08.08
```

After editing the smb.conf file, execute the following steps to enable snapshot access:

1. Run **`service smb [re]start`** to start or restart the **`smb`** service.

2. Enable User Serviceable Snapshot (USS) for Samba. For more information see Section 8.13, “User Serviceable Snapshots”

6.3.8.2. Accessing Snapshot

To access snapshot on the Windows system, execute the following steps:

1. Right Click on the file or directory for which the previous version is required.

2. Click on **Restore previous versions**.

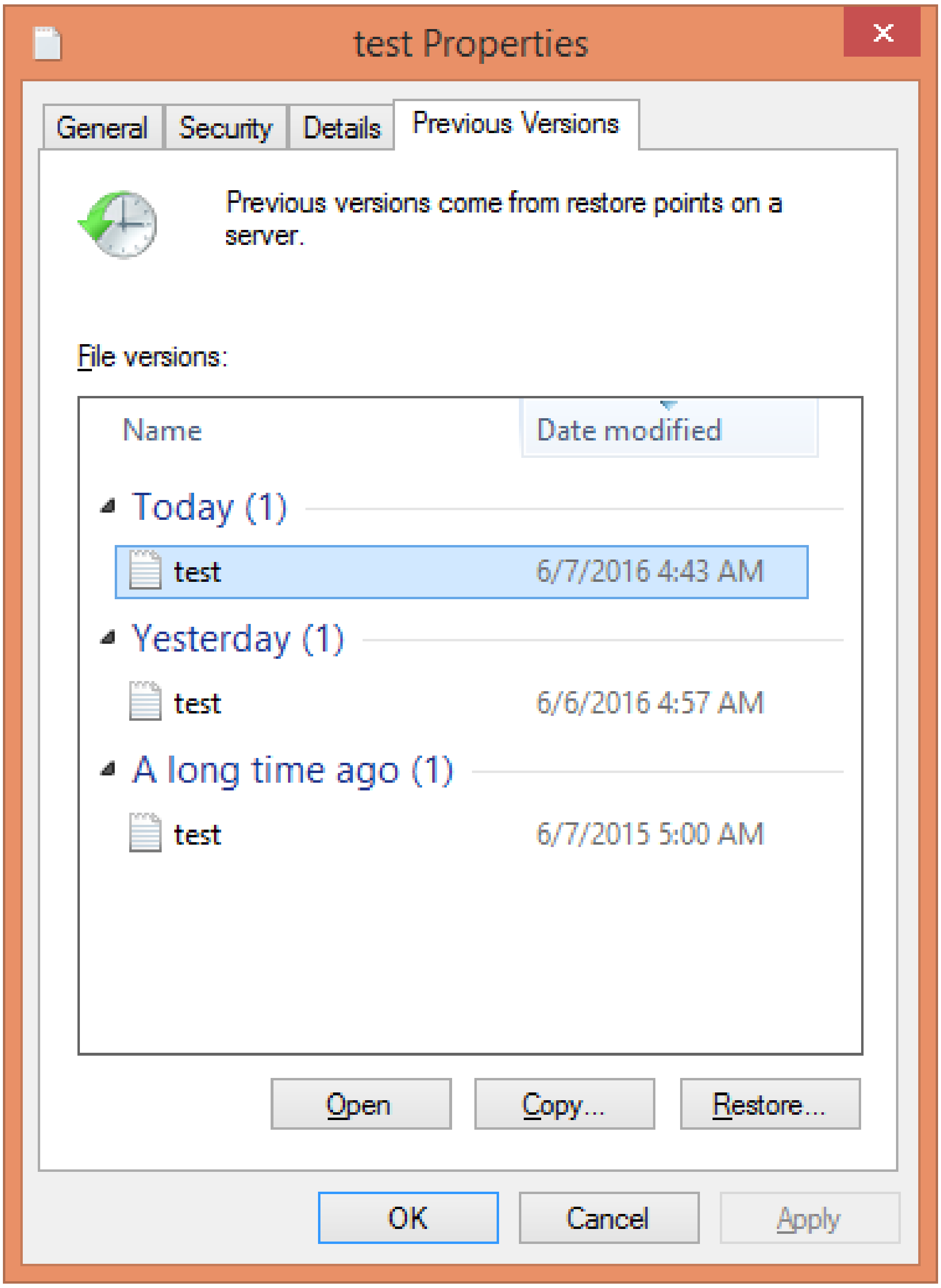
3. In the dialog box, select the Date/Time of the previous version of the file, and select either **Open**, **Restore**, or **Copy**.

   where,

   Open: Lets you open the required version of the file in read-only mode.

   Restore: Restores the file back to the selected version.

   Copy: Lets you copy the file to a different location.



test Properties

Previous versions come from restore points on a server.

File versions:

Name	Date modified
Today (1)	
test	6/7/2016 4:43 AM
Yesterday (1)	
test	6/6/2016 4:57 AM
A long time ago (1)	
test	6/7/2015 5:00 AM

Figure 6.1. Accessing Snapshot

6.3.9. Tuning Performance

This section provides details regarding improving the system performance in an SMB environment. The various enhancements tasks can be classified into:

- Enabling Metadata Caching to improve the performance of SMB access of Red Hat Gluster Storage volumes.
- Enhancing Directory Listing Performance
- Enhancing File/Directory Create Performance

More detailed information for each of this is provided in the sections ahead.

6.3.9.1. Enabling Metadata Caching

Enable metadata caching to improve the performance of directory operations. Execute the following commands from any one of the nodes on the trusted storage pool in the order mentioned below.

Note

If majority of the workload is modifying the same set of files and directories simultaneously from multiple clients, then enabling metadata caching might not provide the desired performance improvement.

1. Execute the following command to enable metadata caching and cache invalidation:

   ```
   # gluster volume set <volname> group metadata-cache
   ```

   This is group set option which sets multiple volume options in a single command.

2. To increase the number of files that can be cached, execute the following command:

   ```
   # gluster volume set <VOLNAME> network.inode-lru-limit <n>
   ```

   n, is set to 50000. It can be increased if the number of active files in the volume is very high. Increasing this number increases the memory footprint of the brick processes.

6.3.9.2. Enhancing Directory Listing Performance

The directory listing gets slower as the number of bricks/nodes increases in a volume, though the file/directory numbers remain unchanged. By enabling the parallel readdir volume option, the performance of directory listing is made independent of the number of nodes/bricks in the volume. Thus, the increase in the scale of the volume does not reduce the directory listing performance.

Note

You can expect an increase in performance only if the distribute count of the volume is 2 or greater and the size of the directory is small (< 3000 entries). The larger the volume (distribute count) greater is the performance benefit.

To enable parallel readdir execute the following commands:

1. Verify if the **`performance.readdir-ahead`** option is enabled by executing the following command:

   ```
   # gluster volume get <VOLNAME> performance.readdir-ahead
   ```

   If the **`performance.readdir-ahead`** is not enabled then execute the following command:

   ```
   # gluster volume set <VOLNAME> performance.readdir-ahead on
   ```

2. Execute the following command to enable **`parallel-readdir`** option:

   ```
   # gluster volume set <VOLNAME> performance.parallel-readdir on
   ```

Note

If there are more than 50 bricks in the volume it is recommended to increase the cache size to be more than 10Mb (default value):

```
# gluster volume set <VOLNAME> performance.rda-cache-limit <CACHE
SIZE>
```

6.3.9.3. Enhancing File/Directory Create Performance

Before creating / renaming any file, lookups (5-6 in SMB) are sent to verify if the file already exists. By serving these lookup from the cache when possible, increases the create / rename performance by multiple folds in SMB access.

1. Execute the following command to enable negative-lookup cache:

```
# gluster volume set <volname> group nl-cache
  volume set success
```

Note

The above command also enables cache-invalidation and increases the timeout to 10 minutes.

6.4. POSIX Access Control Lists

Basic Linux file system permissions are assigned based on three user types: the owning user, members of the owning group, and all other users. POSIX Access Control Lists (ACLs) work around the limitations of this system by allowing administrators to also configure file and directory access permissions based on any user and any group, rather than just the owning user and group.

This section covers how to view and set access control lists, and how to ensure this feature is enabled on your Red Hat Gluster Storage volumes. For more detailed information about how ACLs work, see the *Red Hat Enterprise Linux 7 System Administrator's Guide*: https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/System_Administrators_Guide/ch-Access_Control_Lists.html.

6.4.1. Setting ACLs with setfacl

The **`setfacl`** command lets you modify the ACLs of a specified file or directory. You can add access rules for a file with the **`-m`** subcommand, or remove access rules for a file with the **`-x`** subcommand. The basic syntax is as follows:

```
# setfacl subcommand access_rule file_path
```

The syntax of an access rule depends on which roles need to obey the rule.

Rules for users start with `u:`

```
# setfacl -m u:user:perms file_path
```

For example, **setfacl -m u:fred:rw /mnt/data** gives the user **fred** read and write access to the **/mnt/data** directory.

setfacl -x u::w /works_in_progress/my_presentation.txt prevents all users from writing to the **/works_in_progress/my_presentation.txt** file (except the owning user and members of the owning group, as these are controlled by POSIX).

Rules for groups start with g:

```
# setfacl -m g:group:perms file_path
```

For example, **setfacl -m g:admins:rwx /etc/fstab** gives users in the **admins** group read, write, and execute permissions to the **/etc/fstab** file.

setfacl -x g:newbies:x /mnt/harmful_script.sh prevents users in the **newbies** group from executing **/mnt/harmful_script.sh**.

Rules for other users start with o:

```
# setfacl -m o:perms file_path
```

For example, **setfacl -m o:r /mnt/data/public** gives users without any specific rules about their username or group permission to read files in the **/mnt/data/public directory**.

Rules for setting a maximum access level using an effective rights mask start with m:

```
# setfacl -m m:mask file_path
```

For example, **setfacl -m m:r-x /mount/harmless_script.sh** gives all users a maximum of read and execute access to the **/mount/harmless_script.sh** file.

You can set the default ACLs for a directory by adding **d:** to the beginning of any rule, or make a rule recursive with the **-R** option. For example, **setfacl -Rm d:g:admins:rwx /etc** gives all members of the **admins** group read, write, and execute access to any file created under the **/etc** directory after the point when **setfacl** is run.

6.4.2. Checking current ACLs with getfacl

The **getfacl** command lets you check the current ACLs of a file or directory. The syntax for this command is as follows:

```
# getfacl file_path
```

This prints a summary of current ACLs for that file. For example:

```
# getfacl /mnt/gluster/data/test/sample.jpg
# owner: antony
# group: antony
user::rw-
group::rw-
other::r--
```

If a directory has default ACLs set, these are prefixed with **`default:`**, like so:

```
# getfacl /mnt/gluster/data/doc
# owner: antony
# group: antony
user::rw-
user:john:r--
group::r--
mask::r--
other::r--
default:user::rwx
default:user:antony:rwx
default:group::r-x
default:mask::rwx
default:other::r-x
```

6.4.3. Mounting volumes with ACLs enabled

To mount a volume with ACLs enabled using the Native FUSE Client, use the **`acl`** mount option. For further information, see Section 6.1.3, “Mounting Red Hat Gluster Storage Volumes”.

ACLs are enabled by default on volumes mounted using the NFS and SMB access protocols. To check whether ACLs are enabled on other mounted volumes, see Section 6.4.4, “Checking ACL enablement on a mounted volume”.

6.4.4. Checking ACL enablement on a mounted volume

The following table shows you how to verify that ACLs are enabled on a mounted volume, based on the type of client your volume is mounted with.

Table 6.9.

Client type	How to check	Further info

Client type	How to check	Further info

Native FUSE	Check the output of the **`mount`** command for the **`default_permissions`** option:
	`# mount | grep mountpoint`
	If **`default_permissions`** appears in the output for a mounted volume, ACLs are not enabled on that volume.
	Check the output of the **`ps aux`** command for the gluster FUSE mount process (glusterfs):
	`# ps aux | grep gluster`
	`root      30548  0.0  0.7 548408`
	`13868 ?        Ssl  12:39   0:00`
	`/usr/local/sbin/glusterfs --acl --`
	`volfile-server=127.0.0.2 --`
	`volfile-id=testvol /mnt/fuse_mnt`
	If **`--acl`** appears in the output for a mounted volume, ACLs are enabled on that volume.
		See Section 6.1, “Native Client” for more information.

Gluster Native NFS	On the server side, check the output of the **`gluster volume info`** ***`volname`*** command. If **`nfs.acl`** appears in the output, that volume has ACLs disabled. If **`nfs.acl`** does not appear, ACLs are enabled (the default state). On the client side, check the output of the **`mount`** command for the volume. If **`noacl`** appears in the output, ACLs are disabled on the mount point. If this does not appear in the output, the client checks that the server uses ACLs, and uses ACLs if server support is enabled.	Refer to the output of **`gluster volume set help`** pertaining to NFS, or see the Red Hat Enterprise Linux *Storage Administration Guide* for more information: https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/Storage_Administration_Guide/ch-nfs.html

Client type	How to check	Further info
NFS Ganesha	On the server side, check the volume's export configuration file, **`/run/gluster/shared_storage/nfs-ganesha/exports/export.volname.conf`**. If the **`Disable_ACL`** option is set to **`true`**, ACLs are disabled. Otherwise, ACLs are enabled for that volume. **Note** NFS-Ganesha supports NFSv4 protocol standardized ACLs but not NFSACL protocol used for NFSv3 mounts. Only NFSv4 mounts can set ACLs. There is no option to disable NFSv4 ACLs on the client side, so as long as the server supports ACLs, clients can set ACLs on the mount point.	See Section 6.2.3, “NFS Ganesha” for more information. For client side settings, refer to the Red Hat Enterprise Linux *Storage Administration Guide*: https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/Storage_Administration_Guide/ch-nfs.html
samba	POSIX ACLs are enabled by default when using Samba to access a Red Hat Gluster Storage volume.	See Section 6.3, “SMB” for more information.

6.5. Managing Object Store

Object Store provides a system for data storage that enables users to access the same data, both as an object and as a file, thus simplifying management and controlling storage costs.

Red Hat Gluster Storage is based on glusterFS, an open source distributed file system. Object Store technology is built upon OpenStack Swift. OpenStack Swift allows users to store and retrieve files and content through a simple Web Service REST (Representational State Transfer) interface as objects. Red Hat Gluster Storage uses glusterFS as a back-end file system for OpenStack Swift. It also leverages on OpenStack Swift's REST interface for storing and retrieving files over the web combined with glusterFS features like scalability and high availability, replication, and elastic volume management for data management at disk level.

Object Store technology enables enterprises to adopt and deploy cloud storage solutions. It allows users to access and modify data as objects from a REST interface along with the ability to access and modify files from NAS interfaces. In addition to decreasing cost and making it faster and easier to access object data, it also delivers massive scalability, high availability and replication of object storage. Infrastructure as a Service (IaaS) providers can utilize Object Store technology to enable their own cloud storage service. Enterprises can use this technology to accelerate the process of preparing file-based applications for the cloud and simplify new application development for cloud computing environments.

OpenStack Swift is an open source software for creating redundant, scalable object storage using clusters of standardized servers to store petabytes of accessible data. It is not a file system or real-time data storage system, but rather a long-term storage system for a more permanent type of static data that can be retrieved, leveraged, and updated.

6.5.1. Architecture Overview

OpenStack Swift and Red Hat Gluster Storage integration consists of:

- OpenStack Object Storage environment.

  For detailed information on Object Storage, see OpenStack Object Storage Administration Guide available at: http://docs.openstack.org/admin-guide-cloud/content/ch_admin-openstack-object-storage.html.

- Red Hat Gluster Storage environment.

  Red Hat Gluster Storage environment consists of bricks that are used to build volumes. For more information on bricks and volumes, see Section 5.4, “Formatting and Mounting Bricks”.

The following diagram illustrates OpenStack Object Storage integration with Red Hat Gluster Storage:

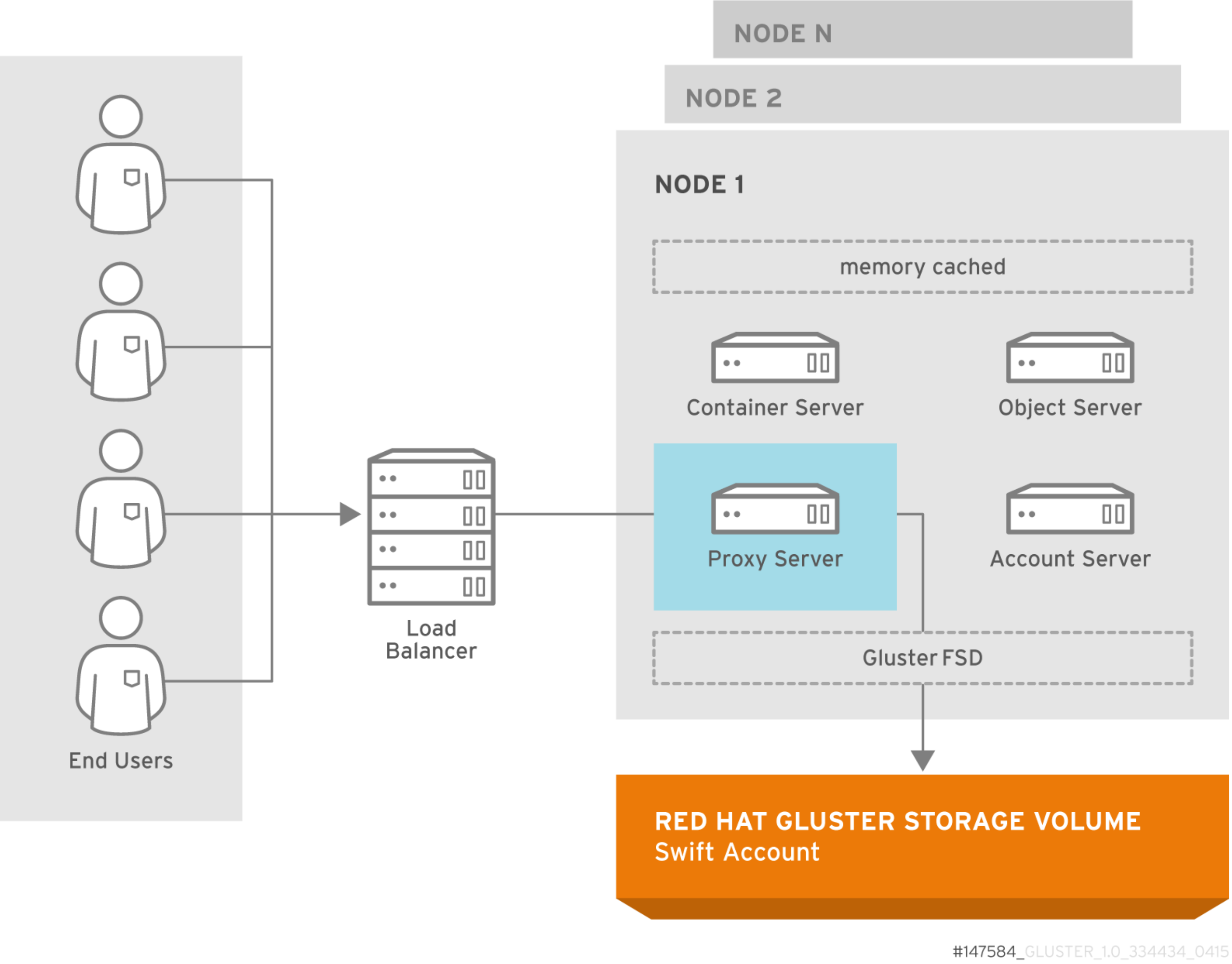


Figure 6.2. Object Store Architecture

Important

On Red Hat Enterprise Linux 7, enable the Object Store firewall service in the active zones for runtime and permanent mode using the following commands:

To get a list of active zones, run the following command:

```
# firewall-cmd  --get-active-zones
```

To add ports to the active zones, run the following commands:

```
# firewall-cmd  --zone=zone_name  --add-port=6010/tcp  --add-
port=6011/tcp --add-port=6012/tcp  --add-port=8080/tcp

# firewall-cmd  --zone=zone_name --add-port=6010/tcp  --add-
port=6011/tcp --add-port=6012/tcp  --add-port=8080/tcp   --permanent
```

Add the port number 443 **only** if your swift proxy server is configured with SSL. To add the port number, run the following commands:

```
# firewall-cmd --zone=zone_name --add-port=443/tcp
# firewall-cmd --zone=zone_name --add-port=443/tcp --permanent
```

6.5.2. Components of Object Store

The major components of Object Storage are:

Proxy Server

The Proxy Server is responsible for connecting to the rest of the OpenStack Object Storage architecture. For each request, it looks up the location of the account, container, or object in the ring and routes the request accordingly. The public API is also exposed through the proxy server. When objects are streamed to or from an object server, they are streamed directly through the proxy server to or from the user – the proxy server does not spool them.

The Ring

The Ring maps swift accounts to the appropriate Red Hat Gluster Storage volume. When other components need to perform any operation on an object, container, or account, they need to interact with the Ring to determine the correct Red Hat Gluster Storage volume.

Object and Object Server

An object is the basic storage entity and any optional metadata that represents the data you store. When you upload data, the data is stored as-is (with no compression or encryption).

The Object Server is a very simple storage server that can store, retrieve, and delete objects stored on local devices.

Container and Container Server

A container is a storage compartment for your data and provides a way for you to organize your data. Containers can be visualized as directories in a Linux system. However, unlike directories, containers cannot be nested. Data must be stored in a container and hence the objects are created within a container.

The Container Server's primary job is to handle listings of objects. The listing is done by querying the glusterFS mount point with a path. This query returns a list of all files and directories present under that container.

Accounts and Account Servers

The OpenStack Swift system is designed to be used by many different storage consumers.

The Account Server is very similar to the Container Server, except that it is responsible for listing containers rather than objects. In Object Store, each Red Hat Gluster Storage volume is an account.

Authentication and Access Permissions

Object Store provides an option of using an authentication service to authenticate and authorize user access. Once the authentication service correctly identifies the user, it will provide a token which must be passed to Object Store for all subsequent container and object operations.

Other than using your own authentication services, the following authentication services are supported by Object Store:

- Authenticate Object Store against an external OpenStack Keystone server.

  Each Red Hat Gluster Storage volume is mapped to a single account. Each account can have multiple users with different privileges based on the group and role they are assigned to. After authenticating using *accountname:username* and *password*, user is issued a token which will be used for all subsequent REST requests.

  ##### Integration with Keystone

  When you integrate Red Hat Gluster Storage Object Store with Keystone authentication, you must ensure that the Swift account name and Red Hat Gluster Storage volume name are the same. It is common that Red Hat Gluster Storage volumes are created before exposing them through the Red Hat Gluster Storage Object Store.

  When working with Keystone, account names are defined by Keystone as the **`tenant id`**. You must create the Red Hat Gluster Storage volume using the Keystone **`tenant id`** as the name of the volume. This means, you must create the Keystone tenant before creating a Red Hat Gluster Storage Volume.

  > **Important**
  >
  > Red Hat Gluster Storage does not contain any Keystone server components. It only acts as a Keystone client. After you create a volume for Keystone, ensure to export this volume for accessing it using the object storage interface. For more information on exporting volume, see Section 6.5.7.8, "Exporting the Red Hat Gluster Storage Volumes".

  ##### Integration with GSwauth

  GSwauth is a Web Server Gateway Interface (WGSI) middleware that uses a Red Hat Gluster Storage Volume itself as its backing store to maintain its metadata. The benefit in this authentication service is to have the metadata available to all proxy servers and saving the data to a Red Hat Gluster Storage volume.

  To protect the metadata, the Red Hat Gluster Storage volume should only be able to be mounted by the systems running the proxy servers. For more information on mounting volumes, see Chapter 6, *Creating Access to Volumes*.

Integration with TempAuth

You can also use the **`TempAuth`** authentication service to test Red Hat Gluster Storage Object Store in the data center.

6.5.3. Advantages of using Object Store

The advantages of using Object Store include:

- Default object size limit of 1 TiB
- Unified view of data across NAS and Object Storage technologies
- High availability
- Scalability
- Replication
- Elastic Volume Management

6.5.4. Limitations

This section lists the limitations of using Red Hat Gluster Storage Object Store:

- Object Name

  Object Store imposes the following constraints on the object name to maintain the compatibility with network file access:

  - Object names must not be prefixed or suffixed by a '/' character. For example, **`a/b/`**
  - Object names must not have contiguous multiple '/' characters. For example, **`a//b`**

- Account Management

  - Object Store does not allow account management even though OpenStack Swift allows the management of accounts. This limitation is because Object Store treats **`accounts`** equivalent to the Red Hat Gluster Storage volumes.
  - Object Store does not support account names (i.e. Red Hat Gluster Storage volume names) having an underscore.
  - In Object Store, every account must map to a Red Hat Gluster Storage volume.

- Subdirectory Listing

  Headers **`X-Content-Type: application/directory`** and **`X-Content-Length: 0`** can be used to create subdirectory objects under a container, but GET request on a subdirectory would not list all the objects under it.

6.5.5. Swift API Support Matrix

Subject to the limitations mentioned in Section 6.5.4, “Limitations”, the following table describes the support status for current Swift API’s functional features:

Table 6.10. Supported Features

Feature	Status
Authentication	Supported
Get Account Metadata	Supported
Swift ACLs	Supported
List Containers	Supported
Delete Container	Supported
Create Container	Supported
Get Container Metadata	Supported
Update Container Metadata	Supported
Delete Container Metadata	Supported
List Objects	Supported
Static Website	Supported
Create/Update an Object	Supported
Create Large Object	Supported
Delete Object	Supported
Get Object	Supported
Copy Object	Supported
Get Object Metadata	Supported
Add/Update Object Metadata	Supported
Temp URL Operations	Supported
Expiring Objects	Supported
Object Versioning	Supported
Cross-Origin Resource Sharing (CORS)	Supported
Bulk Upload	Supported
Account Quota	Unsupported
Container Quota	Unsupported

6.5.6. Prerequisites

Ensure that you do the following before using Red Hat Gluster Storage Object Store.

- Ensure that the *openstack-swift-** and *swiftonfile* packages have matching version numbers.

```
# rpm -qa | grep swift
openstack-swift-container-1.13.1-6.el7ost.noarch
openstack-swift-object-1.13.1-6.el7ost.noarch
swiftonfile-1.13.1-6.el7rhgs.noarch
openstack-swift-proxy-1.13.1-6.el7ost.noarch
openstack-swift-doc-1.13.1-6.el7ost.noarch
openstack-swift-1.13.1-6.el7ost.noarch
openstack-swift-account-1.13.1-6.el7ost.noarch
```

- Ensure that the gluster-swift services are owned by and run as the **`root`** user, not the **`swift`** user as in a typical OpenStack installation.

```
# cd /usr/lib/systemd/system
# sed -i s/User=swift/User=root/ openstack-swift-proxy.service openstack-
swift-account.service openstack-swift-container.service openstack-swift-
object.service openstack-swift-object-expirer.service
```

- Start the **`memcached`** service:

```
# service memcached start
```

- Ensure that the ports for the Object, Container, Account, and Proxy servers are open. Note that the ports used for these servers are configurable. The ports listed in Table 6.11, “Ports required for Red Hat Gluster Storage Object Store” are the default values.

Table 6.11. Ports required for Red Hat Gluster Storage Object Store

Server	Port
Object Server	6010
Container Server	6011
Account Server	6012
Proxy Server (HTTPS)	443
Proxy Server (HTTP)	8080

- Create and mount a Red Hat Gluster Storage volume for use as a Swift Account. For information on creating Red Hat Gluster Storage volumes, see Chapter 5, *Setting Up Storage Volumes* . For information on mounting Red Hat Gluster Storage volumes, see Chapter 6, *Creating Access to Volumes* .

6.5.7. Configuring the Object Store

This section provides instructions on how to configure Object Store in your storage environment.

Warning

When you install Red Hat Gluster Storage 3.2 and higher, the **/etc/swift** directory would contain both ***.conf** extension and ***.conf-gluster** files. You must delete the ***.conf** files and create new configuration files based on ***.conf-gluster** template. Otherwise, inappropriate python packages will be loaded and the component may not work as expected.

If you are upgrading to Red Hat Gluster Storage 3.2 and higher, the older configuration files will be retained and new configuration files will be created with **.rpmnew** extension. You must ensure to delete **.conf** files and folders (account-server, container-server, and object-server) for better understanding of the loaded configuration.

6.5.7.1. Configuring a Proxy Server

Create a new configuration file **/etc/swift/proxy-server.conf** by referencing the template file available at **/etc/swift/proxy-server.conf-gluster**.

6.5.7.1.1. Configuring a Proxy Server for HTTPS

By default, proxy server only handles HTTP requests. To configure the proxy server to process HTTPS requests, perform the following steps:

1. Create self-signed cert for SSL using the following commands:

```
# cd /etc/swift
# openssl req -new -x509 -nodes -out cert.crt -keyout cert.key
```

2. Add the following lines to **/etc/swift/proxy-server.conf** under *[DEFAULT]*

```
bind_port = 443
 cert_file = /etc/swift/cert.crt
 key_file = /etc/swift/cert.key
```

Important

When Object Storage is deployed on two or more machines, not all nodes in your trusted storage pool are used. Installing a load balancer enables you to utilize all the nodes in your trusted storage pool by distributing the proxy server requests equally to all storage nodes.

Memcached allows nodes' states to be shared across multiple proxy servers. Edit the **`memcache_servers`** configuration option in the **`proxy-server.conf`** and list all memcached servers.

Following is an example listing the memcached servers in the **`proxy-server.conf`** file.

```
[filter:cache]
use = egg:swift#memcache
memcache_servers =
192.168.1.20:11211,192.168.1.21:11211,192.168.1.22:11211
```

The port number on which the memcached server is listening is 11211. You must ensure to use the same sequence for all configuration files.

6.5.7.2. Configuring the Authentication Service

This section provides information on configuring **`Keystone, GSwauth,`** and **`TempAuth`** authentication services.

6.5.7.2.1. Integrating with the Keystone Authentication Service

- To configure Keystone, add **`authtoken`** and **`keystoneauth`** to **`/etc/swift/proxy-server.conf`** pipeline as shown below:

  ```
  [pipeline:main]
  pipeline = catch_errors healthcheck proxy-logging cache authtoken
  keystoneauth proxy-logging proxy-server
  ```

- Add the following sections to **`/etc/swift/proxy-server.conf`** file by referencing the example below as a guideline. You must substitute the values according to your setup:

  ```
  [filter:authtoken]
  paste.filter_factory = keystoneclient.middleware.auth_token:filter_factory
  signing_dir = /etc/swift
  auth_host = keystone.server.com
  auth_port = 35357
  auth_protocol = http
  auth_uri = http://keystone.server.com:5000
  # if its defined
  admin_tenant_name = services
  admin_user = swift
  ```

```
admin_password = adminpassword
delay_auth_decision = 1

[filter:keystoneauth]
use = egg:swift#keystoneauth
operator_roles = admin, SwiftOperator
is_admin = true
cache = swift.cache
```

Verify the Integrated Setup

Verify that the Red Hat Gluster Storage Object Store has been configured successfully by running the following command:

```
$ swift -V 2 -A http://keystone.server.com:5000/v2.0 -U tenant_name:user -K password stat
```

6.5.7.2.2. Integrating with the GSwauth Authentication Service

Integrating GSwauth

Perform the following steps to integrate GSwauth:

1. Create and start a Red Hat Gluster Storage volume to store metadata.

   ```
   # gluster volume create NEW-VOLNAME NEW-BRICK
   # gluster volume start NEW-VOLNAME
   ```

   For example:

   ```
   # gluster volume create gsmetadata server1:/rhgs/brick1
   # gluster volume start gsmetadata
   ```

2. Run **`gluster-swift-gen-builders`** tool with all the volumes to be accessed using the Swift client including **`gsmetadata`** volume:

   ```
   # gluster-swift-gen-builders gsmetadata other volumes
   ```

3. Edit the **`/etc/swift/proxy-server.conf`** pipeline as shown below:

   ```
   [pipeline:main]
   pipeline = catch_errors cache gswauth proxy-server
   ```

4. Add the following section to **`/etc/swift/proxy-server.conf`** file by referencing the example below as a guideline. You must substitute the values according to your setup.

   ```
   [filter:gswauth]
   use = egg:gluster_swift#gswauth
   set log_name = gswauth
   super_admin_key = gswauthkey
   metadata_volume = gsmetadata
   auth_type = sha1
   auth_type_salt = swauthsalt
   ```

> **Important**
>
> You must ensure to secure the **proxy-server.conf** file and the **super_admin_key** option to prevent unprivileged access.

5. Restart the proxy server by running the following command:

```
# swift-init proxy restart
```

Advanced Options:

You can set the following advanced options for GSwauth WSGI filter:

- default-swift-cluster: The default storage-URL for the newly created accounts. When you attempt to authenticate for the first time, the access token and the storage-URL where data for the given account is stored will be returned.

- token_life: The set default token life. The default value is 86400 (24 hours).

- max_token_life: The maximum token life. You can set a token lifetime when requesting a new token with header **x-auth-token-lifetime**. If the passed in value is greater than the **max_token_life**, then the **max_token_life** value will be used.

GSwauth Common Options of CLI Tools

GSwauth provides CLI tools to facilitate managing accounts and users. All tools have some options in common:

- -A, --admin-url: The URL to the auth. The default URL is **http://127.0.0.1:8080/auth/**.

- -U, --admin-user: The user with administrator rights to perform action. The default user role is **.super_admin**.

- -K, --admin-key: The key for the user with administrator rights to perform the action. There is no default value.

Preparing Red Hat Gluster Storage Volumes to Save Metadata

Prepare the Red Hat Gluster Storage volume for **gswauth** to save its metadata by running the following command:

```
# gswauth-prep [option]
```

For example:

```
# gswauth-prep -A http://10.20.30.40:8080/auth/ -K gswauthkey
```

6.5.7.2.2.1. Managing Account Services in GSwauth

Creating Accounts

Create an account for GSwauth. This account is mapped to a Red Hat Gluster Storage volume.

```
# gswauth-add-account [option] <account_name>
```

For example:

```
# gswauth-add-account -K gswauthkey <account_name>
```

Deleting an Account

You must ensure that all users pertaining to this account must be deleted before deleting the account. To delete an account:

```
# gswauth-delete-account [option] <account_name>
```

For example:

```
# gswauth-delete-account -K gswauthkey test
```

Setting the Account Service

Sets a service URL for an account. User with **reseller admin** role only can set the service URL. This command can be used to change the default storage URL for a given account. All accounts will have the same storage-URL as default value, which is set using **default-swift-cluster** option.

```
# gswauth-set-account-service [options] <account> <service> <name> <value>
```

For example:

```
# gswauth-set-account-service -K gswauthkey test storage local
http://newhost:8080/v1/AUTH_test
```

6.5.7.2.2.2. Managing User Services in GSwauth

User Roles

The following user roles are supported in GSwauth:

- A regular user has no rights. Users must be given both read and write privileges using Swift ACLs.
- The **admin** user is a super-user at the account level. This user can create and delete users for that account. These members will have both write and read privileges to all stored objects in that account.
- The **reseller admin** user is a super-user at the cluster level. This user can create and delete accounts and users and has read and write privileges to all accounts under that cluster.
- GSwauth maintains its own swift account to store all of its metadata on accounts and users. The **.super_admin** role provides access to GSwauth own swift account and has all privileges to act on any other account or user.

The following table provides user access right information.

Table 6.12. User Role/Group with Allowed Actions

Role/Group	Allowed Actions

Role/Group	Allowed Actions
.super_admin (username)	Get Account List Get Account Details Create Account Delete Account Get User Details Create admin user Create reseller_admin user Create regular user Delete admin user
.reseller_admin (group)	Get Account List Get Account Details Create Account Delete Account Get User Details Create admin user Create regular user Delete admin user
.admin (group)	Get Account Details Get User Details Create admin user Create regular user Delete admin user
regular user (type)	No administrative actions.

Creating Users

You can create an user for an account that does not exist. The account will be created before creating the user.

You must add **`-r`** flag to create a **`reseller admin`** user and **`-a`** flag to create an **`admin`** user. To change the password or role of the user, you can run the same command with the new option.

```
# gswauth-add-user [option] <account_name> <user> <password>
```

For example

```
# gswauth-add-user -K gswauthkey -a test ana anapwd
```

Deleting a User

Delete a user by running the following command:

```
# gswauth-delete-user [option] <account_name> <user>
```

For example

```
# gwauth-delete-user -K gswauthkey test ana
```

Authenticating a User with the Swift Client

There are two methods to access data using the Swift client. The first and simple method is by providing the user name and password everytime. The swift client will acquire the token from gswauth.

For example:

```
$ swift -A http://127.0.0.1:8080/auth/v1.0 -U test:ana -K anapwd upload
container1 README.md
```

The second method is a two-step process, first you must authenticate with a username and password to obtain a token and the storage URL. Then, you can make the object requests to the storage URL with the given token.

It is important to remember that tokens expires, so the authentication process needs to be repeated very often.

Authenticate a user with the cURL command:

```
# curl -v -H 'X-Storage-User: test:ana' -H 'X-Storage-Pass: anapwd' -k
http://localhost:8080/auth/v1.0
...
< X-Auth-Token: AUTH_tk7e68ef4698f14c7f95af07ab7b298610
< X-Storage-Url: http://127.0.0.1:8080/v1/AUTH_test
...
```

Now, you use the given token and storage URL to access the object-storage using the Swift client:

```
$ swift --os-auth-token=AUTH_tk7e68ef4698f14c7f95af07ab7b298610 --os-
storage-url=http://127.0.0.1:8080/v1/AUTH_test upload container1 README.md
README.md
bash-4.2$
bash-4.2$ swift --os-auth-token=AUTH_tk7e68ef4698f14c7f95af07ab7b298610 --
os-storage-url=http://127.0.0.1:8080/v1/AUTH_test list container1
README.md
```

Important

`Reseller admins` must always use the second method to acquire a token to get access to other accounts other than his own. The first method of using the username and password will give them access only to their own accounts.

6.5.7.2.2.3. Managing Accounts and Users Information

Obtaining Accounts and User Information

You can obtain the accounts and users information including stored password.

```
# gswauth-list [options] [account] [user]
```

For example:

```
# gswauth-list -K gswauthkey test ana
```

```
+----------+
|  Groups  |
+----------+
| test:ana |
|   test   |
|  .admin  |
+----------+
```

- If [account] and [user] are omitted, all the accounts will be listed.
- If [account] is included but not [user], a list of users within that account will be listed.
- If [account] and [user] are included, a list of groups that the user belongs to will be listed.
- If the [user] is *.groups*, the active groups for that account will be listed.

The default output format is in tabular format. Adding **`-p`** option provides the output in plain text format, **`-j`** provides the output in JSON format.

Changing User Password

You can change the password of the user, account administrator, and reseller_admin roles.

- Change the password of a regular user by running the following command:

  ```
  # gswauth-add-user -U account1:user1 -K old_passwd account1 user1
  new_passwd
  ```

- Change the password of an **`account administrator`** by running the following command:

  ```
  # gswauth-add-user -U account1:admin -K old_passwd -a account1 admin
  new_passwd
  ```

- Change the password of the **`reseller_admin`** by running the following command:

  ```
  # gswauth-add-user -U account1:radmin -K old_passwd -r account1 radmin
  new_passwd
  ```

Cleaning Up Expired Tokens

Users with **`.super_admin`** role can delete the expired tokens.

You also have the option to provide the expected life of tokens, delete all tokens or delete all tokens for a given account.

```
# gswauth-cleanup-tokens [options]
```

For example

```
# gswauth-cleanup-tokens -K gswauthkey --purge test
```

The tokens will be deleted on the disk but it would still persist in memcached.

You can add the following options while cleaning up the tokens:

- -t, --token-life: The expected life of tokens. The token objects modified before the give number of seconds will be checked for expiration (default: 86400).

- --purge: Purges all the tokens for a given account whether the tokens have expired or not.
- --purge-all: Purges all the tokens for all the accounts and users whether the tokens have expired or not.

6.5.7.2.3. Integrating with the TempAuth Authentication Service

Warning

TempAuth authentication service must only be used in test deployments and not for production.

TempAuth is automatically installed when you install Red Hat Gluster Storage. TempAuth stores user and password information as **cleartext** in a single **proxy-server.conf** file. In your **/etc/swift/proxy-server.conf** file, enable TempAuth in pipeline and add user information in **TempAuth** section by referencing the below example.

```
[pipeline:main]
pipeline = catch_errors healthcheck proxy-logging cache tempauth proxy-
logging proxy-server

[filter:tempauth]
use = egg:swift#tempauth
user_admin_admin = admin.admin.reseller_admin
user_test_tester = testing .admin
user_test_tester2 = testing2
```

You can add users to the account in the following format:

```
user_accountname_username = password [.admin]
```

Here the **accountname** is the Red Hat Gluster Storage volume used to store objects.

You must restart the Object Store services for the configuration changes to take effect. For information on restarting the services, see Section 6.5.7.9, “Starting and Stopping Server”.

6.5.7.3. Configuring Object Servers

Create a new configuration file **/etc/swift/object.server.conf** by referencing the template file available at **/etc/swift/object-server.conf-gluster**.

6.5.7.4. Configuring Container Servers

Create a new configuration file **/etc/swift/container-server.conf** by referencing the template file available at **/etc/swift/container-server.conf-gluster**.

6.5.7.5. Configuring Account Servers

Create a new configuration file **/etc/swift/account-server.conf** by referencing the template file available at **/etc/swift/account-server.conf-gluster**.

6.5.7.6. Configuring Swift Object and Container Constraints

Create a new configuration file **`/etc/swift/swift.conf`** by referencing the template file available at /**`etc/swift/swift.conf-gluster`**.

6.5.7.7. Configuring Object Expiration

The Object Expiration feature allows you to schedule automatic deletion of objects that are stored in the Red Hat Gluster Storage volume. You can use the object expiration feature to specify a lifetime for specific objects in the volume; when the lifetime of an object expires, the object store would automatically quit serving that object and would shortly thereafter remove the object from the Red Hat Gluster Storage volume. For example, you might upload logs periodically to the volume, and you might need to retain those logs for only a specific amount of time.

The client uses the X-Delete-At or X-Delete-After headers during an object PUT or POST and the Red Hat Gluster Storage volume would automatically quit serving that object.

Note

Expired objects appear in container listings until they are deleted by the **`object-expirer`** daemon. This is an expected behavior.

A DELETE object request on an expired object would delete the object from Red Hat Gluster Storage volume (if it is yet to be deleted by the object expirer daemon). However, the client would get a 404 (Not Found) status in return. This is also an expected behavior.

6.5.7.7.1. Setting Up Object Expiration

Object expirer uses a separate account (a Red Hat Gluster Storage volume) named **`gsexpiring`** for managing object expiration. Hence, you must create a Red Hat Gluster Storage volume and name it as **`gsexpiring`**.

Create a new configuration file **`/etc/swift/object.expirer.conf`** by referencing the template file available at **`/etc/swift/object-expirer.conf-gluster`**.

6.5.7.7.2. Using Object Expiration

When you use the X-Delete-At or X-Delete-After headers during an object PUT or POST, the object is scheduled for deletion. The Red Hat Gluster Storage volume would automatically quit serving that object at the specified time and will shortly thereafter remove the object from the Red Hat Gluster Storage volume.

Use PUT operation while uploading a new object. To assign expiration headers to existing objects, use the POST operation.

X-Delete-At header

The X-Delete-At header requires a UNIX epoch timestamp, in integer form. For example, 1418884120 represents Thu, 18 Dec 2014 06:27:31 GMT. By setting the header to a specific epoch time, you indicate when you want the object to expire, not be served, and be deleted completely from the Red Hat Gluster Storage volume. The current time in Epoch notation can be found by running this command:

```
$ date +%s
```

- Set the object expiry time during an object PUT with X-Delete-At header using cURL:

```
# curl -v -X PUT -H 'X-Delete-At: 1392013619'
http://127.0.0.1:8080/v1/AUTH_test/container1/object1 -T ./localfile
```

Set the object expiry time during an object PUT with X-Delete-At header using swift client:

```
# swift --os-auth-token=AUTH_tk99a39aecc3dd4f80b2b1e801d00df846 --os-
storage-url=http://127.0.0.1:8080/v1/AUTH_test upload container1
./localfile --header 'X-Delete-At: 1392013619'
```

X-Delete-After

The X-Delete-After header takes an integer number of seconds that represents the amount of time from now when you want the object to be deleted.

- Set the object expiry time with an object PUT with X-Delete-After header using cURL:

```
# curl -v -X PUT -H 'X-Delete-After: 3600'
http://127.0.0.1:8080/v1/AUTH_test/container1/object1 -T ./localfile
```

Set the object expiry time with an object PUT with X-Delete-At header using swift client:

```
# swift --os-auth-token=AUTH_tk99a39aecc3dd4f80b2b1e801d00df846 --os-
storage-url=http://127.0.0.1:8080/v1/AUTH_test upload container1
./localfile --header 'X-Delete-After: 3600'
```

6.5.7.7.3. Running Object Expirer Service

The object-expirer service runs once in every 300 seconds, by default. You can modify the duration by configuring **interval** option in **/etc/swift/object-expirer.conf** file. For every pass it makes, it queries the gsexpiring account for **tracker objects**. Based on the timestamp and path present in the name of **tracker objects**, object-expirer deletes the actual object and the corresponding tracker object.

To start the object-expirer service:

```
# swift-init object-expirer start
```

To run the object-expirer once:

```
# swift-object-expirer -o -v /etc/swift/object-expirer.conf
```

6.5.7.8. Exporting the Red Hat Gluster Storage Volumes

After creating configuration files, you must now add configuration details for the system to identify the Red Hat Gluster Storage volumes to be accessible as Object Store. These configuration details are added to the ring files. The ring files provide the list of Red Hat Gluster Storage volumes to be accessible using the object storage interface to the **Swift on File** component.

Create the ring files for the current configurations by running the following command:

```
# cd /etc/swift
# gluster-swift-gen-builders VOLUME [VOLUME...]
```

For example,

```
# cd /etc/swift
# gluster-swift-gen-builders testvol1 testvol2 testvol3
```

Here *testvol1, testvol2, and testvol3* are the Red Hat Gluster Storage volumes which will be mounted locally under the directory mentioned in the object, container, and account configuration files (default value is **`/mnt/gluster-object`**). The default value can be changed to a different path by changing the **`devices`** configurable option across all account, container, and object configuration files. The path must contain Red Hat Gluster Storage volumes mounted under directories having the same names as volume names. For example, if **`devices`** option is set to **`/home`**, it is expected that the volume named **`testvol1`** be mounted at **`/home/testvol1`**.

Note that all the volumes required to be accessed using the Swift interface must be passed to the **`gluster-swift-gen-builders`** tool even if it was previously added. The **`gluster-swift-gen-builders`** tool creates new ring files every time it runs successfully.

To remove a *VOLUME*, run **`gluster-swift-gen-builders`** only with the volumes which are required to be accessed using the Swift interface.

For example, to remove the **`testvol2`** volume, run the following command:

```
# gluster-swift-gen-builders testvol1 testvol3
```

You must restart the Object Store services after creating the new ring files.

6.5.7.9. Starting and Stopping Server

You must start or restart the server manually whenever you update or modify the configuration files. These processes must be owned and run by the root user.

- To start the server, run the following command:

  ```
  # swift-init main start
  ```

- To stop the server, run the following command:

  ```
  # swift-init main stop
  ```

- To restart the server, run the following command:

  ```
  # swift-init main restart
  ```

6.5.8. Starting the Services Automatically

To configure the gluster-swift services to start automatically when the system boots, run the following commands:

On Red Hat Enterprise Linux 6:

```
# chkconfig memcached on
# chkconfig openstack-swift-proxy on
# chkconfig openstack-swift-account on
# chkconfig openstack-swift-container on
# chkconfig openstack-swift-object on
# chkconfig openstack-swift-object-expirer on
```

On Red Hat Enterprise Linux 7:

```
# systemctl enable openstack-swift-proxy.service
# systemctl enable openstack-swift-account.service
# systemctl enable openstack-swift-container.service
# systemctl enable openstack-swift-object.service
# systemctl enable openstack-swift-object-expirer.service
# systemctl enable openstack-swift-object-expirer.service
```

Configuring the gluster-swift services to start at boot time by using the **`systemctl`** command may require additional configuration. Refer to https://access.redhat.com/solutions/2043773 for details if you encounter problems.

> **Important**
>
> You must restart all Object Store services servers whenever you change the configuration and ring files.

6.5.9. Working with the Object Store

For more information on Swift operations, see OpenStack Object Storage API Reference Guide available at http://docs.openstack.org/api/openstack-object-storage/1.0/content/ .

6.5.9.1. Creating Containers and Objects

Creating container and objects in Red Hat Gluster Storage Object Store is very similar to OpenStack swift. For more information on Swift operations, see OpenStack Object Storage API Reference Guide available at http://docs.openstack.org/api/openstack-object-storage/1.0/content/.

6.5.9.2. Creating Subdirectory under Containers

You can create a subdirectory object under a container using the headers **`Content-Type: application/directory`** and **`Content-Length: 0`**. However, the current behavior of Object Store returns **`200 OK`** on a **`GET`** request on subdirectory but this does not list all the objects under that subdirectory.

6.5.9.3. Working with Swift ACLs

Swift ACLs work with users and accounts. ACLs are set at the container level and support lists for read and write access. For more information on Swift ACLs, see http://docs.openstack.org/user-guide/content/managing-openstack-object-storage-with-swift-cli.html.

6.6. Checking Client Operating Versions

Different versions of Red Hat Gluster Storage support different features. Servers and clients identify the features that they are capable of supporting using an operating version number, or **`op-version`**. The ***`cluster.op-version`*** parameter sets the required operating version for all volumes in a cluster on the server side. Each client supports a range of operating versions that are identified by a minimum (***`min-op-version`***) and maximum (***`max-op-version`***) supported operating version.

Check the operating versions of the clients connected to a given volume by running the following command:

For Red Hat Gluster 3.2 and later

```
# gluster volume status volname clients
```

Use **`all`** in place of the name of your volume if you want to see the operating versions of clients connected to all volumes in the cluster.

Before Red Hat Gluster Storage 3.2:

1. Perform a state dump for the volume whose clients you want to check.

   ```
   # gluster volume statedump volname
   ```

2. Locate the state dump directory

   ```
   # gluster --print-statedumpdir
   ```

3. Locate the state dump file and grep for client information.

   ```
   # grep -A4 "identifier=client_ip" statedumpfile
   ```

Chapter 7. Integrating Red Hat Gluster Storage with Windows Active Directory

In this chapter, the tasks necessary for integrating Red Hat Gluster Storage nodes into an existing Windows Active Directory domain are described. The following diagram describes the architecture of integrating Red Hat Gluster Storage with Windows Active Directory.

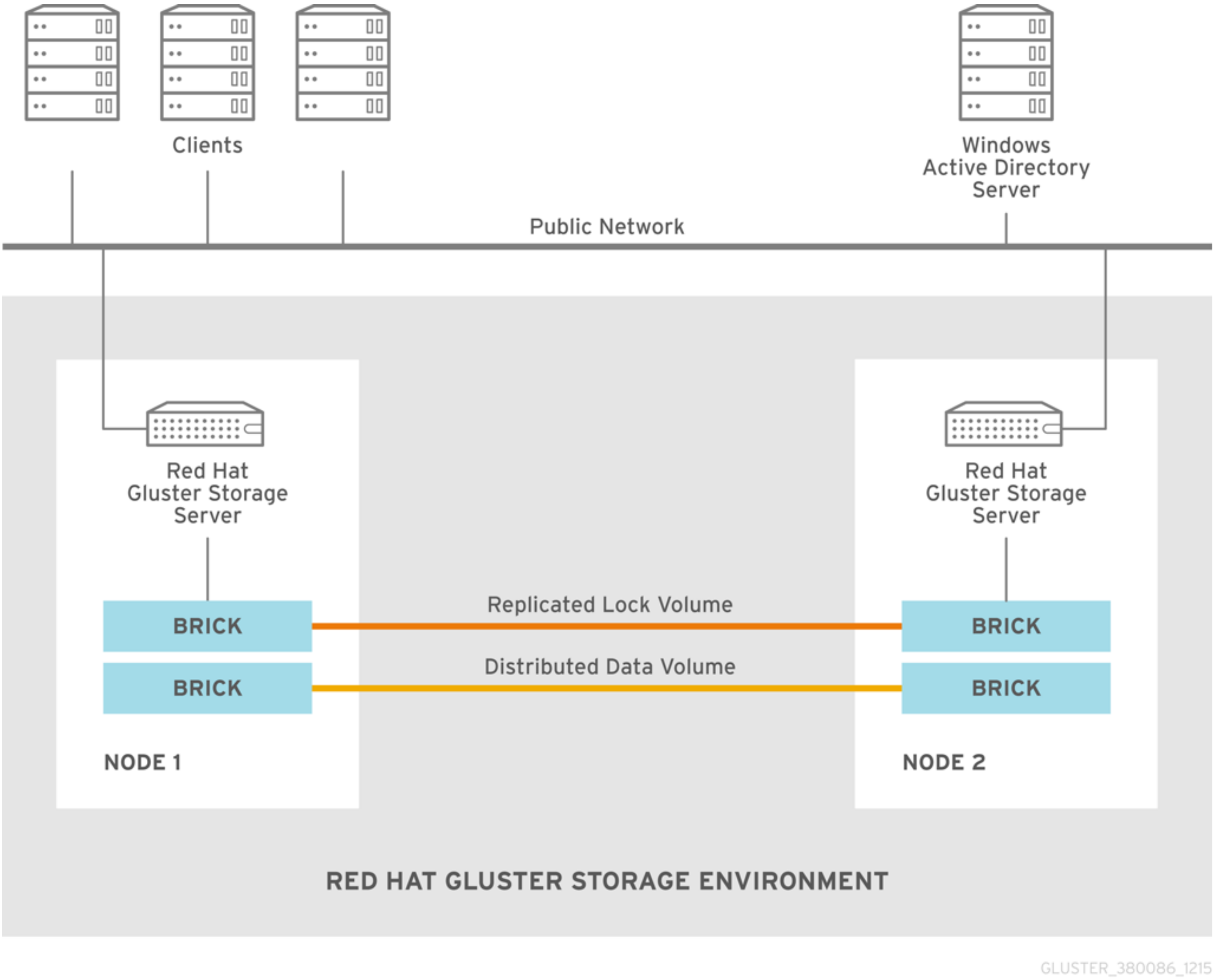

Clients
Windows Active Directory Server
Public Network
Red Hat Gluster Storage Server
Red Hat Gluster Storage Server
Replicated Lock Volume
Distributed Data Volume
BRICK
BRICK
BRICK
BRICK
NODE 1
NODE 2
RED HAT GLUSTER STORAGE ENVIRONMENT
GLUSTER_380086_1215

Figure 7.1. Active Directory Integration

This section assumes that you have an active directory domain installed. Before we go ahead with the configuration details, following is a list of data along with examples that will be used in the sections ahead.

Table 7.1.

Information	Example Value
DNS domain name / realm	addom.example.com
NetBIOS domain name	ADDOM
Name of administrative account	administrator
RHGS nodes	rhs-srv1.addom.example.com, 192.168.56.10 rhs-srv2.addom.example.com, 192.168.56.11 rhs-srv3.addom.example.com, 192.168.56.12
Netbios name of the cluster	RHS-SMB

7.1. Prerequisites

Before integration, the following steps have to be completed on an existing Red Hat Gluster Storage environment:

- ### Name Resolution

  The Red Hat Gluster Storage nodes must be able to resolve names from the AD domain via DNS. To verify the same you can use the following command:

  ```
  host dc1.addom.example.com
  ```

  where, **addom.example.com** is the AD domain and dc1 is the name of a domain controller.

  For example, the **/etc/resolv.conf** file in a static network configuration could look like this:

  ```
  domain addom.example.com
  search addom.example.com
  nameserver 10.11.12.1 # dc1.addom.example.com
  nameserver 10.11.12.2 # dc2.addom.example.com
  ```

  This example assumes that both the domain controllers are also the DNS servers of the domain.

- ### Kerberos Packages

  If you want to use the kerberos client utilities, like kinit and klist, then manually install the krb5-workstation using the following command:

  ```
  # yum -y install krb5-workstation
  ```

- ### Synchronize Time Service

  It is essential that the time service on each Red Hat Gluster Storage node and the Windows Active Directory server are synchronized, else the Kerberos authentication may fail due to clock skew. In environments where time services are not reliable, the best practice is to configure the Red Hat Gluster Storage nodes to synchronize time from the Windows Server.

  On each Red Hat Storage node, edit the file /etc/ntp.conf so the time is synchronized from a known, reliable time service:

  ```
  # Enable writing of statistics records.
  #statistics clockstats cryptostats loopstats peerstats
  server ntp1.addom.example.com
  server 10.11.12.3
  ```

  Activate the change on each Red Hat Gluster Storage node by stopping the ntp daemon, updating the time, then starting the ntp daemon. Verify the change on both servers using the following commands:

  ```
  # service ntpd stop

  # service ntpd start
  ```

- ### Samba Packages

Ensure to install the following Samba packages along with its dependencies:

- CTDB
- samba
- samba-client
- samba-winbind
- samba-winbind-modules

7.2. Integration

Integrating Red Hat Gluster Storage Servers into an Active Directory domain involves the following series of steps:

1. Configure Authentication
2. Join Active Directory Domain
3. Verify/Test Active Directory and Services

7.2.1. Configure Authentication

In order to join a cluster to the Active Directory domain, a couple of files have to be edited manually on all nodes.

Note

- Ensure that CTDB is configured before the active directory join. For more information see, *Section 6.3.1 Setting up CTDB for Samba* in the *Red Hat Gluster Storage Administration Guide*.
- It is recommended to take backups of the configuration and of Samba's databases (local and ctdb) before making any changes.

7.2.1.1. Basic Samba Configuration

As of Red Hat Gluster Storage 3.4 Batch 4 Update, the recommended idmap configuration method for new deployments is **`autorid`**, not **`tdb`**. Red Hat recommends **`autorid`** because in addition to automatically calculating user and group identifiers like **`tdb`**, it performs fewer database transactions and read operations, and is a prerequisite for supporting secure ID history (SID history).

Warning

Do not change the idmap configuration in existing deployments. Doing so requires a large number of changes, such as modifying the permissions and access control lists of all files in the shared file system, which unless done carefully can create user access problems. If you do need to change the idmap configuration settings for an existing deployment, contact Red Hat support for assistance.

The Samba configuration file **/etc/samba/smb.conf** must be identical on all nodes, and must contain the relevant parameters for AD. Along with that, a few other settings are required in order to activate mapping of user and group IDs.

The following example depicts the minimal Samba configuration for AD integration:

```
[global]
netbios name = RHS-SMB
workgroup = ADDOM
realm = addom.example.com
security = ads
clustering = yes
idmap config * : backend = autorid
idmap config * : range = 1000000-19999999
idmap config * : rangesize = 1000000

# -----------------RHS Options -------------------------
#
# The following line includes RHS-specific configuration options. Be careful
with this line.

       include = /etc/samba/rhs-samba.conf

#=================Share Definitions =====================
```

Warning

The example above is the complete **global** section required in the **smb.conf** file. Ensure that nothing else appears in this section in order to prevent gluster mechanisms from changing settings when starting or stopping the ctdb lock volume.

The ***netbios name*** consists of only one name which has to be the same name on all cluster nodes. Windows clients will only access the cluster via that name (either in this short form or as an FQDN). The individual node hostname (rhs-srv1, rhs-srv2, …) must not be used for the **netbios name** parameter.

Note

- The idmap ***range*** defines the lowerst and hightest identifier numbers that can be used. Specify a range large enough to cover the number of objects specified in ***rangesize***.
- The idmap ***rangesize*** specifies the number of identifiers available for each domain range. In this case there are one million identifiers per domain range, and the ***range*** parameter indicates that there are nearly 19 million identifiers total, meaning that there are a total of 19 possible domain ranges.
- If you want to be able to use the individual host names to also access specific nodes, you can add them to the ***netbios aliases*** parameter of **smb.conf**.
- In an AD environment, it is usually not required to run **nmbd**. However, if you have to run **nmbd**, then make sure to set the **cluster addresses** **smb.conf** option to the list of public IP addresses of the cluster.

7.2.1.2. Alternative Configuration using ad backend

If you need full control over Active Directory IDs, you can adapt the Samba configuration further by using the **idmap_ad** module in addition to **autorid**. The **idmap_ad** module reads the unix IDs from the AD's special unix attributes. This has to be configured by the AD domain's administrator before it can be used by Samba and winbind.

In order for Samba to use **idmap_ad**, the AD domain admin has to prepare the AD domain for using the so called unix extensions and assign unix IDs to all users and groups that should be able to access the Samba server.

For example, following is an extended Samba configuration file to use the **idmap_ad** backend for the ADDOM domain. The default **autorid** backend catches all objects from domains other than the ADDOM domain.

```
[global]
netbios name = RHS-SMB
workgroup = ADDOM
realm = addom.example.com
security = ads
clustering = yes
idmap config * : backend = autorid
idmap config * : range = 1000000-1999999
idmap config ADDOM : backend = ad
idmap config ADDOM : range = 3000000-3999999
idmap config ADDOM : schema mode = rfc2307
winbind nss info = rfc2307

# -------------------RHS Options -------------------------------
#
# The following line includes RHS-specific configuration options. Be careful
with this line.

       include = /etc/samba/rhs-samba.conf

#===================Share Definitions =========================
```

Note

- The range for the idmap_ad configuration is prescribed by the AD configuration. This has to be obtained by AD administrator.
- Ranges for different idmap configurations must not overlap.
- The schema mode and the winbind nss info setting should have the same value. If the domain is at level 2003R2 or newer, then rfc2307 is the correct value. For older domains, additional values sfu and sfu20 are available. See the manual pages of idmap_ad and smb.conf for further details.

7.2.1.3. Verifying the Samba Configuration

Test the new configuration file using the testparm command. For example:

```
# testparm -s
Load smb config files from /etc/samba/smb.conf
```

```
rlimit_max: increasing rlimit_max (1024) to minimum Windows limit (16384)
Loaded services file OK.

Server role: ROLE_DOMAIN_MEMBER

# Global parameters
[global]
    workgroup = ADDOM
    realm = addom.example.com
    netbios name = RHS-SMB
    security = ADS
    clustering = Yes
    winbind nss info = rfc2307
    idmap config addom : schema mode = rfc2307
    idmap config addom : range = 3000000-3999999
    idmap config addom : backend = ad
    idmap config * : range = 1000000-1999999
    idmap config * : backend = autorid
```

7.2.1.4. nsswitch Configuration

Once the Samba configuration has been made, Samba has to be enabled to use the mapped users and groups from AD. This is achieved via the local Name Service Switch (NSS) that has to be made aware of the winbind. To use the winbind NSS module, edit the **`/etc/nsswitch.conf`** file. Make sure the file contains the winbind entries for the **`passwd`** and **`group`** databases. For example:

```
...
passwd: files winbind
group: files winbind
...
```

This will enable the use of winbind and should make users and groups **`visible`** on the individual cluster node once Samba is joined to AD and winbind is started.

7.2.2. Join Active Directory Domain

Prior to joining AD, CTDB must be started so that the machine account information can be stored in a database file that is available on all cluster nodes via CTDB. In addition to that, all other Samba services should be stopped. If key-based SSH authentication without a password has been configured for the root user between the nodes, you can use the onnode tool to run these commands on all nodes from a single node:

```
# onnode all service ctdb start
# onnode all service winbind stop
# onnode all service smb stop
```

Note

- If your configuration has CTDB managing Winbind and Samba, they can be temporarily disabled with the following commands (to be executed prior to the above stop commands) so as to prevent CTDB going into an unhealthy state when they are shut down:

  ```
  # onnode all ctdb event script disable 49.winbind
  # onnode all ctdb event script disable 50.samba
  ```

- For some versions of RHGS, a bug in the selinux policy prevents 'ctdb disablescript SCRIPT' from succeeding. If this is the case, 'chmod -x /etc/ctdb/events.d/SCRIPT' can be executed as a workaround from a root shell.
- Shutting down winbind and smb is primarily to prevent access to SMB services during this AD integration. These services may be left running but access to them should be prevented through some other means.

The join is initiated via the **net** utility from a single node:

Warning

The following step must be executed only on one cluster node and should not be repeated on other cluster nodes. CTDB makes sure that the whole cluster is joined by this step.

```
# net ads join -U Administrator
Enter Administrator's password:
Using short domain name -- ADDOM
Joined 'RHS-SMB' to dns domain addom.example.com'
Not doing automatic DNS update in a clustered setup.
```

Once the join is successful, the cluster ip addresses and the cluster netbios name should be made public in the network. For registering multiple public cluster IP addresses in the AD DNS server, the **net** utility can be used again:

```
# net ads dns register rhs-smb <PUBLIC IP 1> <PUBLIC IP 2> ...
```

This command will make sure the DNS name **rhs-smb** will resolve to the given public IP addresses. The DNS registrations use the cluster machine account for authentication in AD, which means this operation only can be done after the join has succeeded.

Registering the NetBIOS name of the cluster is done by the nmbd service. In order to make sure that the nmbd instances on the hosts don’t overwrite each other’s registrations, the ‘cluster addresses’ smb.conf option should be set to the list of public addresses of the whole cluster.

7.2.3. Verify/Test Active Directory and Services

When the join is successful, the Samba and the Winbind daemons can be started.

Start nmdb using the following command:

```
# onnode all service nmb start
```

Start the winbind and smb services:

```
# onnode all service winbind start
# onnode all service smb start
```

Note

- If you previously disabled CTDB’s ability to manage Winbind and Samba they can be re-enabled with the following commands:

  ```
  # onnode all ctdb event script enable 50.samba
  # onnode all ctdb event script enable 49.winbind
  ```

- For some versions of RHGS, a bug in the selinux polict prevents 'ctdb enablescript SCRIPT' from succeeding. If this is the case, 'chmod +x /etc/ctdb/events.d/SCRIPT' can be executed as a workaround from a root shell.
- Ensure that the winbind starts after a reboot. This is achieved by adding ‘CTDB_MANAGES_WINBIND=yes’ to the /etc/sysconfig/ctdb file on all nodes.

Execute the following verification steps:

1. **Verify the join by executing the following steps**

   Verify the join to check if the created machine account can be used to authenticate to the AD LDAP server using the following command:

   ```
   # net ads testjoin
   Join is OK
   ```

2. Execute the following command to display the machine account’s LDAP object

   ```
   # net ads status -P
   objectClass: top
   objectClass: person
   objectClass: organizationalPerson
   objectClass: user
   objectClass: computer
   cn: rhs-smb
   distinguishedName: CN=rhs-smb,CN=Computers,DC=addom,DC=example,DC=com
   instanceType: 4
   whenCreated: 20150922013713.0Z
   whenChanged: 20151126111120.0Z
   displayName: RHS-SMB$
   uSNCreated: 221763
   uSNChanged: 324438
   name: rhs-smb
   objectGUID: a178177e-4aa4-4abc-9079-d1577e137723
   userAccountControl: 69632
   ```

```
badPwdCount: 0
codePage: 0
countryCode: 0
badPasswordTime: 130880426605312806
lastLogoff: 0
lastLogon: 130930100623392945
localPolicyFlags: 0
pwdLastSet: 130930098809021309
primaryGroupID: 515
objectSid: S-1-5-21-2562125317-1564930587-1029132327-1196
accountExpires: 9223372036854775807
logonCount: 1821
sAMAccountName: rhs-smb$
sAMAccountType: 805306369
dNSHostName: rhs-smb.addom.example.com
servicePrincipalName: HOST/rhs-smb.addom.example.com
servicePrincipalName: HOST/RHS-SMB
objectCategory:
CN=Computer,CN=Schema,CN=Configuration,DC=addom,DC=example,DC=com
isCriticalSystemObject: FALSE
dSCorePropagationData: 16010101000000.0Z
lastLogonTimestamp: 130929563322279307
msDS-SupportedEncryptionTypes: 31
```

3. Execute the following command to display general information about the AD server:

```
# net ads info
LDAP server: 10.11.12.1
LDAP server name: dc1.addom.example.com
Realm: ADDOM.EXAMPLE.COM
Bind Path: dc=ADDOM,dc=EXAMPLE,dc=COM
LDAP port: 389
Server time: Thu, 26 Nov 2015 11:15:04 UTC
KDC server: 10.11.12.1
Server time offset: -26
```

4. **Verify if winbind is operating correctly by executing the following steps**

   Execute the following command to verify if winbindd can use the machine account for authentication to AD

```
# wbinfo -t
checking the trust secret for domain ADDOM via RPC calls succeeded
```

5. Execute the following command to resolve the given name to a Windows SID

```
# wbinfo --name-to-sid 'ADDOM\Administrator'
S-1-5-21-2562125317-1564930587-1029132327-500 SID_USER (1)
```

6. Execute the following command to verify authentication:

```
# wbinfo -a 'ADDOM\user'
Enter ADDOM\user's password:
plaintext password authentication succeeded
```

```
Enter ADDOM\user's password:
challenge/response password authentication succeeded
```

or,

```
# wbinfo -a 'ADDOM\user%password'
plaintext password authentication succeeded
challenge/response password authentication succeeded
```

7. Execute the following command to verify if the id-mapping is working properly:

```
# wbinfo --sid-to-uid <SID-OF-ADMIN>
1000000
```

8. Execute the following command to verify if the winbind Name Service Switch module works correctly:

```
# getent passwd 'ADDOM\Administrator'
ADDOM\administrator:*:1000000:1000004::/home/ADDOM/administrator:/bin/
false
```

9. Execute the following command to verify if samba can use winbind and the NSS module correctly:

```
# smbclient -L rhs-smb -U 'ADDOM\Administrator'
Domain=[ADDOM] OS=[Windows 6.1] Server=[Samba 4.2.4]

        Sharename       Type      Comment
        ---------       ----      -------
        IPC$            IPC       IPC Service (Samba 4.2.4)
Domain=[ADDOM] OS=[Windows 6.1] Server=[Samba 4.2.4]

        Server               Comment
        ---------            -------
        RHS-SMB         Samba 4.2.4

        Workgroup            Master
        ---------            -------
        ADDOM              RHS-SMB
```

Part IV. Manage

Chapter 8. Managing Snapshots

Red Hat Gluster Storage Snapshot feature enables you to create point-in-time copies of Red Hat Gluster Storage volumes, which you can use to protect data. Users can directly access Snapshot copies which are read-only to recover from accidental deletion, corruption, or modification of the data.

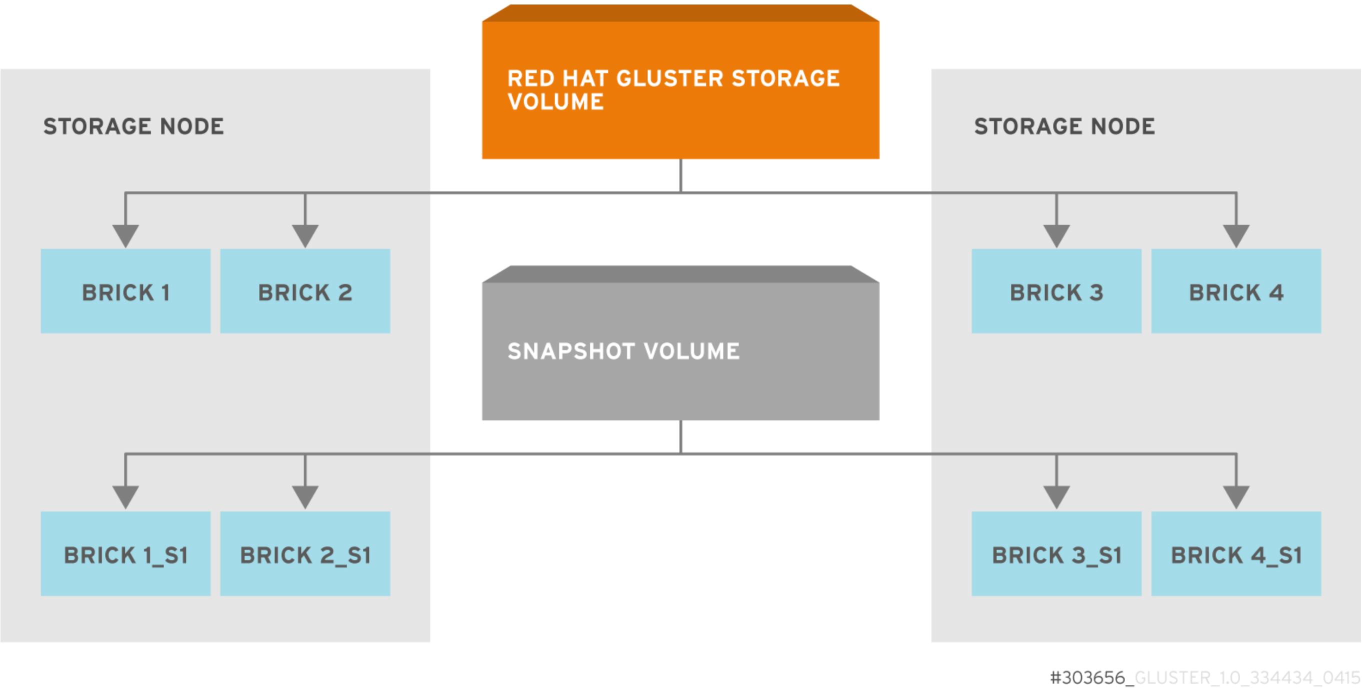


Figure 8.1. Snapshot Architecture

In the Snapshot Architecture diagram, Red Hat Gluster Storage volume consists of multiple bricks (Brick1 Brick2 etc) which is spread across one or more nodes and each brick is made up of independent thin Logical Volumes (LV). When a snapshot of a volume is taken, it takes the snapshot of the LV and creates another brick. Brick1_s1 is an identical image of Brick1. Similarly, identical images of each brick is created and these newly created bricks combine together to form a snapshot volume.

Some features of snapshot are:

- **Crash Consistency**

  A crash consistent snapshot is captured at a particular point-in-time. When a crash consistent snapshot is restored, the data is identical as it was at the time of taking a snapshot.

  > **Note**
  >
  > Currently, application level consistency is not supported.

- **Online Snapshot**

  Snapshot is an online snapshot hence the file system and its associated data continue to be available for the clients even while the snapshot is being taken.

- **Quorum Based**

The quorum feature ensures that the volume is in a good condition while the bricks are down. If any brick that is down for a n way replication, where n <= 2 , quorum is not met. In a n-way replication where n >= 3, quorum is met when m bricks are up, where m >= (n/2 +1) where n is odd and m >= n/2 and the first brick is up where n is even. If quorum is not met snapshot creation fails.

- **Barrier**

  To guarantee crash consistency some of the fops are blocked during a snapshot operation.

  These fops are blocked till the snapshot is complete. All other fops is passed through. There is a default time-out of 2 minutes, within that time if snapshot is not complete then these fops are unbarriered. If the barrier is unbarriered before the snapshot is complete then the snapshot operation fails. This is to ensure that the snapshot is in a consistent state.

> **Note**
>
> Taking a snapshot of a Red Hat Gluster Storage volume that is hosting the Virtual Machine Images is not recommended. Taking a Hypervisor assisted snapshot of a virtual machine would be more suitable in this use case.

8.1. Prerequisites

Before using this feature, ensure that the following prerequisites are met:

- Snapshot is based on thinly provisioned LVM. Ensure the volume is based on LVM2. Red Hat Gluster Storage is supported on Red Hat Enterprise Linux 6.7 and later and Red Hat Enterprise Linux 7.1 and later. Both these versions of Red Hat Enterprise Linux is based on LVM2 by default. For more information, see https://access.redhat.com/site/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Logical_Volume_Manager_Administration/thinprovisioned_volumes.html

- Each brick must be independent thinly provisioned logical volume(LV).

- The logical volume which contains the brick must not contain any data other than the brick.

- Only linear LVM is supported with Red Hat Gluster Storage. For more information, see https://access.redhat.com/site/documentation/en-US/Red_Hat_Enterprise_Linux/4/html-single/Cluster_Logical_Volume_Manager/#lv_overview

Recommended Setup

The recommended setup for using Snapshot is described below. In addition, you must ensure to read Chapter 21, *Tuning for Performance* for enhancing snapshot performance:

- For each volume brick, create a dedicated thin pool that contains the brick of the volume and its (thin) brick snapshots. With the current thin-p design, avoid placing the bricks of different Red Hat Gluster Storage volumes in the same thin pool, as this reduces the performance of snapshot operations, such as snapshot delete, on other unrelated volumes.

- The recommended thin pool chunk size is 256KB. There might be exceptions to this in cases where we have a detailed information of the customer's workload.

- The recommended pool metadata size is 0.1% of the thin pool size for a chunk size of 256KB or larger. In special cases, where we recommend a chunk size less than 256KB, use a pool metadata size of 0.5% of thin pool size.

For Example

To create a brick from device /dev/sda1.

1. Create a physical volume(PV) by using the **`pvcreate`** command.

   ```
   pvcreate /dev/sda1
   ```

   Use the correct **`dataalignment`** option based on your device. For more information, Section 21.2, “Brick Configuration”

2. Create a Volume Group (VG) from the PV using the following command:

   ```
   vgcreate dummyvg /dev/sda1
   ```

3. Create a thin-pool using the following command:

   ```
   # lvcreate --size 1T --thin dummyvg/dummypool --chunksize 256k --
   poolmetadatasize 16G  --zero n
   ```

   A thin pool of size 1 TB is created, using a chunksize of 256 KB. Maximum pool metadata size of 16 G is used.

4. Create a thinly provisioned volume from the previously created pool using the following command:

   ```
   # lvcreate --virtualsize 1G --thin dummyvg/dummypool --name dummylv
   ```

5. Create a file system (XFS) on this. Use the recommended options to create the XFS file system on the thin LV.

   For example,

   ```
   mkfs.xfs -f -i size=512 -n size=8192 /dev/dummyvg/dummylv
   ```

6. Mount this logical volume and use the mount path as the brick.

   ```
   mount /dev/dummyvg/dummylv /mnt/brick1
   ```

8.2. Creating Snapshots

Before creating a snapshot ensure that the following prerequisites are met:

- Red Hat Gluster Storage volume has to be present and the volume has to be in the **`Started`** state.
- All the bricks of the volume have to be on an independent thin logical volume(LV).
- Snapshot names must be unique in the cluster.
- All the bricks of the volume should be up and running, unless it is a n-way replication where n >= 3. In such case quorum must be met. For more information see Chapter 8, *Managing Snapshots*
- No other volume operation, like **`rebalance`**, **`add-brick`**, etc, should be running on the volume.
- Total number of snapshots in the volume should not be equal to *Effective snap-max-hard-limit*. For more information see *Configuring Snapshot Behavior*.

- If you have a geo-replication setup, then pause the geo-replication session if it is running, by executing the following command:

  ```
  # gluster volume geo-replication MASTER_VOL SLAVE_HOST::SLAVE_VOL pause
  ```

  For example,

  ```
  # gluster volume geo-replication master-vol example.com::slave-vol pause
  Pausing geo-replication session between master-vol example.com::slave-vol
  has been successful
  ```

  Ensure that you take the snapshot of the master volume and then take snapshot of the slave volume.

To create a snapshot of the volume, run the following command:

```
# gluster snapshot create <snapname> <volname> [no-timestamp] [description
<description>] [force]
```

where,

- *snapname* - Name of the snapshot that will be created.
- *VOLNAME(S)* - Name of the volume for which the snapshot will be created. We only support creating snapshot of single volume.
- *description* - This is an optional field that can be used to provide a description of the snap that will be saved along with the snap.
- **`force`** - Snapshot creation will fail if any brick is down. In a n-way replicated Red Hat Gluster Storage volume where n >= 3 snapshot is allowed even if some of the bricks are down. In such case quorum is checked. Quorum is checked only when the **`force`** option is provided, else by-default the snapshot create will fail if any brick is down. Refer the *Overview* section for more details on quorum.
- no-timestamp: By default a timestamp is appended to the snapshot name. If you do not want to append timestamp then pass no-timestamp as an argument.

Note

Snapshots are not activated on creation by default; to enable this behavior for all future snapshot creations, set the ***`activate-on-create`*** parameter to **`enabled`**.

For Example 1:

```
# gluster snapshot create snap1 vol1 no-timestamp
snapshot create: success: Snap snap1 created successfully
```

For Example 2:

```
# gluster snapshot create snap1 vol1
snapshot create: success: Snap snap1_GMT-2015.07.20-10.02.33 created
successfully
```

Snapshot of a Red Hat Gluster Storage volume creates a read-only Red Hat Gluster Storage volume. This volume will have identical configuration as of the original / parent volume. Bricks of this newly created snapshot is mounted as **/var/run/gluster/snaps/*<snap-volume-name>*/brick*<bricknumber>***.

For example, a snapshot with snap volume name **0888649a92ea45db8c00a615dfc5ea35** and having two bricks will have the following two mount points:

```
/var/run/gluster/snaps/0888649a92ea45db8c00a615dfc5ea35/brick1
/var/run/gluster/snaps/0888649a92ea45db8c00a615dfc5ea35/brick2
```

These mounts can also be viewed using the **df** or **mount** command.

Note

If you have a geo-replication setup, after creating the snapshot, resume the geo-replication session by running the following command:

```
# gluster volume geo-replication MASTER_VOL SLAVE_HOST::SLAVE_VOL
resume
```

For example,

```
# gluster volume geo-replication master-vol example.com::slave-vol
resume
Resuming geo-replication session between master-vol example.com::slave-
vol has been successful
```

Volume snapshot creation results in the creation of snapshot pool of blocks that contains a copy of the LVM metadata. After taking a snapshot, when new data is written to gluster volume, the snapshot pool is overwritten and the changes are copied to the main gluster volume. As a result, the snapshot pool consumes more metadata space if data changes after the snapshot is taken.

8.3. Cloning a Snapshot

A clone or a writable snapshot is a new volume, which is created from a particular snapshot.

To clone a snapshot, execute the following command.

```
# gluster snapshot clone <clonename> <snapname>
```

where,

clonename: It is the name of the clone, ie, the new volume that will be created.

snapname: It is the name of the snapshot that is being cloned.

Note

- Unlike restoring a snapshot, the original snapshot is still retained, after it has been cloned.
- The snapshot should be in activated state and all the snapshot bricks should be in running state before taking clone. Also the server nodes should be in quorum.
- This is a space efficient clone therefore both the Clone (new volume) and the snapshot LVM share the same LVM backend. The space consumption of the LVM grow as the new volume (clone) diverge from the snapshot.

For example:

```
# gluster snapshot clone clone_vol snap1
snapshot clone: success: Clone clone_vol created successfully
```

To check the status of the newly cloned snapshot execute the following command

```
# gluster vol info <clonename>
```

For example:

```
# gluster vol info clone_vol

Volume Name: clone_vol
Type: Distribute
Volume ID: cdd59995-9811-4348-8e8d-988720db3ab9
Status: Created
Number of Bricks: 1
Transport-type: tcp
Bricks:
Brick1: 10.00.00.01:/var/run/gluster/snaps/clone_vol/brick1/brick3
Options Reconfigured:
performance.readdir-ahead: on
```

In the example it is observed that clone is in **`Created`** state, similar to a newly created volume. This volume should be explicitly started to use this volume.

8.4. Listing of Available Snapshots

To list all the snapshots that are taken for a specific volume, run the following command:

```
# gluster snapshot list [VOLNAME]
```

where,

- *VOLNAME* - This is an optional field and if provided lists the snapshot names of all snapshots present in the volume.

For Example:

```
# gluster snapshot list
snap3
# gluster snapshot list test_vol
No snapshots present
```

8.5. Getting Information of all the Available Snapshots

The following command provides the basic information of all the snapshots taken. By default the information of all the snapshots in the cluster is displayed:

```
# gluster snapshot info [(<snapname> | volume VOLNAME)]
```

where,

- *snapname* - This is an optional field. If the *snapname* is provided then the information about the specified snap is displayed.
- *VOLNAME* - This is an optional field. If the *VOLNAME* is provided the information about all the snaps in the specified volume is displayed.

For Example:

```
# gluster snapshot info snap3
Snapshot                  : snap3
Snap UUID                 : b2a391ce-f511-478f-83b7-1f6ae80612c8
Created                   : 2014-06-13 09:40:57
Snap Volumes:

     Snap Volume Name          : e4a8f4b70a0b44e6a8bff5da7df48a4d
     Origin Volume name        : test_vol1
     Snaps taken for test_vol1      : 1
     Snaps available for test_vol1  : 255
     Status                    : Started
```

8.6. Getting the Status of Available Snapshots

This command displays the running status of the snapshot. By default the status of all the snapshots in the cluster are displayed. To check the status of all the snapshots that are taken for a particular volume, specify a volume name:

```
# gluster snapshot status [(<snapname> | volume VOLNAME)]
```

where,

- *snapname* - This is an optional field. If the *snapname* is provided then the status about the specified snap is displayed.
- *VOLNAME* - This is an optional field. If the *VOLNAME* is provided the status about all the snaps in the specified volume is displayed.

For example:

```
# gluster snapshot status snap3
```

```
Snap Name : snap3
Snap UUID : b2a391ce-f511-478f-83b7-1f6ae80612c8

     Brick Path        :
10.70.42.248:/var/run/gluster/snaps/e4a8f4b70a0b44e6a8bff5da7df48a4d/brick1/
brick1
     Volume Group      :   snap_lvgrp1
     Brick Running     :   Yes
     Brick PID         :   1640
     Data Percentage   :   1.54
     LV Size           :   616.00m


     Brick Path        :
10.70.43.139:/var/run/gluster/snaps/e4a8f4b70a0b44e6a8bff5da7df48a4d/brick2/
brick3
     Volume Group      :   snap_lvgrp1
     Brick Running     :   Yes
     Brick PID         :   3900
     Data Percentage   :   1.80
     LV Size           :   616.00m


     Brick Path        :
10.70.43.34:/var/run/gluster/snaps/e4a8f4b70a0b44e6a8bff5da7df48a4d/brick3/b
rick4
     Volume Group      :   snap_lvgrp1
     Brick Running     :   Yes
     Brick PID         :   3507
     Data Percentage   :   1.80
     LV Size           :   616.00m
```

Note

This shows the status of an activated snapshot.

8.7. Configuring Snapshot Behavior

The configurable parameters for snapshot are:

- **`snap-max-hard-limit`**: If the snapshot count in a volume reaches this limit then no further snapshot creation is allowed. The range is from 1 to 256. Once this limit is reached you have to remove the snapshots to create further snapshots. This limit can be set for the system or per volume. If both system limit and volume limit is configured then the effective max limit would be the lowest of the two value.

- **`snap-max-soft-limit`**: This is a percentage value. The default value is 90%. This configuration works along with auto-delete feature. If auto-delete is enabled then it will delete the oldest snapshot when snapshot count in a volume crosses this limit. When auto-delete is disabled it will not delete any snapshot, but it will display a warning message to the user.

- **`auto-delete`**: This will enable or disable auto-delete feature. By default auto-delete is disabled. When enabled it will delete the oldest snapshot when snapshot count in a volume crosses the snap-max-soft-limit. When disabled it will not delete any snapshot, but it will display a warning message to the user

- **`activate-on-create`**: Snapshots are not activated at creation time by default. If you want created snapshots to immediately be activated after creation, set the ***`activate-on-create`*** parameter to **`enabled`**. Note that all volumes are affected by this setting.

- **Displaying the Configuration Values**

  To display the existing configuration values for a volume or the entire cluster, run the following command:

  ```
  # gluster snapshot config [VOLNAME]
  ```

  where:

  - *VOLNAME*: This is an optional field. The name of the volume for which the configuration values are to be displayed.

  If the volume name is not provided then the configuration values of all the volume is displayed. System configuration details are displayed irrespective of whether the volume name is specified or not.

  For Example:

  ```
  # gluster snapshot config

  Snapshot System Configuration:
  snap-max-hard-limit : 256
  snap-max-soft-limit : 90%
  auto-delete : disable
  activate-on-create : disable

  Snapshot Volume Configuration:

  Volume : test_vol
  snap-max-hard-limit : 256
  Effective snap-max-hard-limit : 256
  Effective snap-max-soft-limit : 230 (90%)

  Volume : test_vol1
  snap-max-hard-limit : 256
  Effective snap-max-hard-limit : 256
  Effective snap-max-soft-limit : 230 (90%)
  ```

- **Changing the Configuration Values**

  To change the existing configuration values, run the following command:

  ```
  # gluster snapshot config [VOLNAME] ([snap-max-hard-limit <count>] [snap-
  max-soft-limit <percent>]) | ([auto-delete <enable|disable>]) |
  ([activate-on-create <enable|disable>])
  ```

  where:

- *VOLNAME*: This is an optional field. The name of the volume for which the configuration values are to be changed. If the volume name is not provided, then running the command will set or change the system limit.
- *snap-max-hard-limit*: Maximum hard limit for the system or the specified volume.
- *snap-max-soft-limit*: Soft limit mark for the system.
- *auto-delete*: This enables or disables the auto-delete feature. By default auto-delete is disabled.
- *activate-on-create*: This enables or disables the activate-on-create feature for all volumes. By default activate-on-create is disabled.

For Example:

```
# gluster snapshot config test_vol snap-max-hard-limit 100
Changing snapshot-max-hard-limit will lead to deletion of snapshots if
they exceed the new limit.
Do you want to continue? (y/n) y
snapshot config: snap-max-hard-limit for test_vol set successfully
```

8.8. Activating and Deactivating a Snapshot

Only activated snapshots are accessible. Check the *Accessing Snapshot* section for more details. Since each snapshot is a Red Hat Gluster Storage volume it consumes some resources hence if the snapshots are not needed it would be good to deactivate them and activate them when required. To activate a snapshot run the following command:

```
# gluster snapshot activate <snapname> [force]
```

where:

- *snapname*: Name of the snap to be activated.
- **force**: If some of the bricks of the snapshot volume are down then use the **force** command to start them.

For Example:

```
# gluster snapshot activate snap1
```

To deactivate a snapshot, run the following command:

```
# gluster snapshot deactivate <snapname>
```

where:

- *snapname*: Name of the snap to be deactivated.

For example:

```
# gluster snapshot deactivate snap1
```

8.9. Deleting Snapshot

Before deleting a snapshot ensure that the following prerequisites are met:

- Snapshot with the specified name should be present.
- Red Hat Gluster Storage nodes should be in quorum.
- No volume operation (e.g. add-brick, rebalance, etc) should be running on the original / parent volume of the snapshot.

To delete a snapshot run the following command:

```
# gluster snapshot delete <snapname>
```

where,

- *snapname* - The name of the snapshot to be deleted.

For Example:

```
# gluster snapshot delete snap2
Deleting snap will erase all the information about the snap. Do you still
want to continue? (y/n) y
snapshot delete: snap2: snap removed successfully
```

Note

Red Hat Gluster Storage volume cannot be deleted if any snapshot is associated with the volume. You must delete all the snapshots before issuing a volume delete.

8.9.1. Deleting Multiple Snapshots

Multiple snapshots can be deleted using either of the following two commands.

To delete all the snapshots present in a system, execute the following command:

```
# gluster snapshot delete all
```

To delete all the snapshot present in a specified volume, execute the following command:

```
# gluster snapshot delete volume <volname>
```

8.10. Restoring Snapshot

Before restoring a snapshot ensure that the following prerequisites are met

- The specified snapshot has to be present
- The original / parent volume of the snapshot has to be in a stopped state.
- Red Hat Gluster Storage nodes have to be in quorum.
- No volume operation (e.g. add-brick, rebalance, etc) should be running on the origin or parent volume of the snapshot.

```
# gluster snapshot restore <snapname>
```

where,

- *snapname* - The name of the snapshot to be restored.

For Example:

```
# gluster snapshot restore snap1
Snapshot restore: snap1: Snap restored successfully
```

After snapshot is restored and the volume is started, trigger a self-heal by running the following command:

```
# gluster volume heal VOLNAME full
```

Note

- The snapshot will be deleted once it is restored. To restore to the same point again take a snapshot explicitly after restoring the snapshot.
- After restore the brick path of the original volume will change. If you are using **`fstab`** to mount the bricks of the origin volume then you have to fix **`fstab`** entries after restore. For more information see, https://access.redhat.com/site/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Installation_Guide/apcs04s07.html

- In the cluster, identify the nodes participating in the snapshot with the snapshot status command. For example:

```
 # gluster snapshot status snapname

    Snap Name : snapname
    Snap UUID : bded7c02-8119-491b-a7e1-cc8177a5a1cd

     Brick Path        :
10.70.43.46:/var/run/gluster/snaps/816e8403874f43a78296decd7c127205/brick2
/brick2
     Volume Group      :   snap_lvgrp
     Brick Running     :   Yes
     Brick PID         :   8303
     Data Percentage   :   0.43
     LV Size           :   2.60g


     Brick Path        :
10.70.42.33:/var/run/gluster/snaps/816e8403874f43a78296decd7c127205/brick3
/brick3
     Volume Group      :   snap_lvgrp
     Brick Running     :   Yes
     Brick PID         :   4594
     Data Percentage   :   42.63
     LV Size           :   2.60g


     Brick Path        :
```

```
10.70.42.34:/var/run/gluster/snaps/816e8403874f43a78296decd7c127205/brick4
/brick4
     Volume Group       :   snap_lvgrp
     Brick Running      :   Yes
     Brick PID          :   23557
     Data Percentage    :   12.41
     LV Size            :   2.60g
```

- In the nodes identified above, check if the **`geo-replication`** repository is present in **`/var/lib/glusterd/snaps/`*`snapname`***. If the repository is present in any of the nodes, ensure that the same is present in **`/var/lib/glusterd/snaps/`*`snapname`*** throughout the cluster. If the **`geo-replication`** repository is missing in any of the nodes in the cluster, copy it to **`/var/lib/glusterd/snaps/`*`snapname`*** in that node.

- Restore snapshot of the volume using the following command:

  ```
  # gluster snapshot restore snapname
  ```

Restoring Snapshot of a Geo-replication Volume

If you have a geo-replication setup, then perform the following steps to restore snapshot:

1. Stop the geo-replication session.

   ```
   # gluster volume geo-replication MASTER_VOL SLAVE_HOST::SLAVE_VOL stop
   ```

2. Stop the slave volume and then the master volume.

   ```
   # gluster volume stop VOLNAME
   ```

3. Restore snapshot of the slave volume and the master volume.

   ```
   # gluster snapshot restore snapname
   ```

4. Start the slave volume first and then the master volume.

   ```
   # gluster volume start VOLNAME
   ```

5. Start the geo-replication session.

   ```
   # gluster volume geo-replication MASTER_VOL SLAVE_HOST::SLAVE_VOL
   start
   ```

6. Resume the geo-replication session.

   ```
   # gluster volume geo-replication MASTER_VOL SLAVE_HOST::SLAVE_VOL
   resume
   ```

8.11. Accessing Snapshots

Snapshot of a Red Hat Gluster Storage volume can be accessed only via FUSE mount. Use the following command to mount the snapshot.

```
mount -t glusterfs <hostname>:/snaps/<snapname>/parent-VOLNAME /mount_point
```

- *parent-VOLNAME* - Volume name for which we have created the snapshot.

  For example,

  ```
  # mount -t glusterfs myhostname:/snaps/snap1/test_vol /mnt
  ```

Since the Red Hat Gluster Storage snapshot volume is read-only, no write operations are allowed on this mount. After mounting the snapshot the entire snapshot content can then be accessed in a read-only mode.

Note

NFS and CIFS mount of snapshot volume is not supported.

Snapshots can also be accessed via User Serviceable Snapshots. For more information see, Section 8.13, "User Serviceable Snapshots"

Warning

External snapshots, such as snapshots of a virtual machine/instance, where Red Hat Gluster Storage Server is installed as a guest OS or FC/iSCSI SAN snapshots are not supported.

8.12. Scheduling of Snapshots

Snapshot scheduler creates snapshots automatically based on the configured scheduled interval of time. The snapshots can be created every hour, a particular day of the month, particular month, or a particular day of the week based on the configured time interval. The following sections describes scheduling of snapshots in detail.

8.12.1. Prerequisites

- To initialize snapshot scheduler on all the nodes of the cluster, execute the following command:

  ```
  snap_scheduler.py init
  ```

  This command initializes the snap_scheduler and interfaces it with the crond running on the local node. This is the first step, before executing any scheduling related commands from a node.

  Note

  This command has to be run on all the nodes participating in the scheduling. Other options can be run independently from any node, where initialization has been successfully completed.

- A shared storage named **`gluster_shared_storage`** is used across nodes to co-ordinate the scheduling operations. This shared storage is mounted at /var/run/gluster/shared_storage on all the nodes. For more information see, Section 11.12, "Setting up Shared Storage Volume"

- All nodes in the cluster have their times synced using NTP or any other mechanism. This is a hard requirement for this feature to work.

- If you are on Red Hat Enterprise Linux 7.1 or later, set the **`cron_system_cronjob_use_shares`** boolean to **on** by running the following command:

  ```
  # setsebool -P cron_system_cronjob_use_shares on
  ```

8.12.2. Snapshot Scheduler Options

Note

There is a latency of one minute, between providing a command by the helper script and for the command to take effect. Hence, currently, we do not support snapshot schedules with per minute granularity.

Enabling Snapshot Scheduler

To enable snap scheduler, execute the following command:

```
snap_scheduler.py enable
```

Note

Snapshot scheduler is disabled by default after initialization

For example:

```
# snap_scheduler.py enable
snap_scheduler: Snapshot scheduling is enabled
```

Disabling Snapshot Scheduler

To enable snap scheduler, execute the following command:

```
 snap_scheduler.py disable
```

For example:

```
# snap_scheduler.py disable
snap_scheduler: Snapshot scheduling is disabled
```

Displaying the Status of Snapshot Scheduler

To display the the current status(Enabled/Disabled) of the snap scheduler, execute the following command:

```
snap_scheduler.py status
```

For example:

```
# snap_scheduler.py status
snap_scheduler: Snapshot scheduling status: Disabled
```

Adding a Snapshot Schedule

To add a snapshot schedule, execute the following command:

```
snap_scheduler.py add "Job Name" "Schedule" "Volume Name"
```

where,

Job Name: This name uniquely identifies this particular schedule, and can be used to reference this schedule for future events like edit/delete. If a schedule already exists for the specified Job Name, the add command will fail.

Schedule: The schedules are accepted in the format crond understands. For example:

```
Example of job definition:
.---------------- minute (0 - 59)
| .------------- hour (0 - 23)
| | .---------- day of month (1 - 31)
| | | .------- month (1 - 12) OR jan,feb,mar,apr ...
| | | | .---- day of week (0 - 6) (Sunday=0 or 7) OR
sun,mon,tue,wed,thu,fri,sat
| | | | |
* * * * * user-name command to be executed
```

Volume name: The name of the volume on which the scheduled snapshot operation will be performed

For example:

```
# snap_scheduler.py add "Job1" "* * * * *" test_vol
snap_scheduler: Successfully added snapshot schedule
```

Note

The snapshots taken by the scheduler will have the following naming convention: Scheduler-<Job Name>-<volume name>_<Timestamp>.

For example:

```
Scheduled-Job1-test_vol_GMT-2015.06.19-09.47.01
```

Editing a Snapshot Schedule

To edit an existing snapshot schedule, execute the following command:

```
snap_scheduler.py edit "Job Name" "Schedule" "Volume Name"
```

where,

Job Name: This name uniquely identifies this particular schedule, and can be used to reference this schedule for future events like edit/delete. If a schedule already exists for the specified Job Name, the add command will fail.

Schedule: The schedules are accepted in the format crond understands. For example:

```
Example of job definition:
.---------------- minute (0 - 59)
| .------------- hour (0 - 23)
| | .---------- day of month (1 - 31)
| | | .------- month (1 - 12) OR jan,feb,mar,apr ...
| | | | .---- day of week (0 - 6) (Sunday=0 or 7) OR
sun,mon,tue,wed,thu,fri,sat
| | | | |
* * * * * user-name command to be executed
```

Volume name: The name of the volume on which the snapshot schedule will be edited.

For Example:

```
# snap_scheduler.py edit "Job1" "*/5 * * * *" gluster_shared_storage
snap_scheduler: Successfully edited snapshot schedule
```

Listing a Snapshot Schedule

To list the existing snapshot schedule, execute the following command:

```
snap_scheduler.py list
```

For example:

```
# snap_scheduler.py list
JOB_NAME         SCHEDULE         OPERATION        VOLUME NAME
--------------------------------------------------------------------
Job0                          * * * * *                Snapshot Create
test_vol
```

Deleting a Snapshot Schedule

To delete an existing snapshot schedule, execute the following command:

```
snap_scheduler.py delete "Job Name"
```

where,

Job Name: This name uniquely identifies the particular schedule that has to be deleted.

For example:

```
# snap_scheduler.py delete Job1
snap_scheduler: Successfully deleted snapshot schedule
```

8.13. User Serviceable Snapshots

User Serviceable Snapshot is a quick and easy way to access data stored in snapshotted volumes. This feature is based on the core snapshot feature in Red Hat Gluster Storage. With User Serviceable Snapshot feature, you can access the activated snapshots of the snapshot volume.

Consider a scenario where a user wants to access a file **`test.txt`** which was in the Home directory a couple of months earlier and was deleted accidentally. You can now easily go to the virtual **`.snaps`** directory that is inside the home directory and recover the test.txt file using the **`cp`** command.

Note

- User Serviceable Snapshot is not the recommended option for bulk data access from an earlier snapshot volume. For such scenarios it is recommended to mount the Snapshot volume and then access the data. For more information see, Chapter 8, *Managing Snapshots*
- Each activated snapshot volume when initialized by User Serviceable Snapshots, consumes some memory. Most of the memory is consumed by various house keeping structures of gfapi and xlators like DHT, AFR, etc. Therefore, the total memory consumption by snapshot depends on the number of bricks as well. Each brick consumes approximately 10MB of space, for example, in a 4x2 replica setup the total memory consumed by snapshot is around 50MB and for a 6x2 setup it is roughly 90MB.

  Therefore, as the number of active snapshots grow, the total memory footprint of the snapshot daemon (snapd) also grows. Therefore, in a low memory system, the snapshot daemon can get **`OOM`** killed if there are too many active snapshots

8.13.1. Enabling and Disabling User Serviceable Snapshot

To enable user serviceable snapshot, run the following command:

```
# gluster volume set VOLNAME features.uss enable
```

For example:

```
# gluster volume set test_vol features.uss enable
volume set: success
```

To disable user serviceable snapshot run the following command:

```
# gluster volume set VOLNAME features.uss disable
```

For example:

```
# gluster volume set test_vol features.uss disable
volume set: success
```

8.13.2. Viewing and Retrieving Snapshots using NFS / FUSE

For every snapshot available for a volume, any user who has access to the volume will have a read-only view of the volume. You can recover the files through these read-only views of the volume from different point in time. Each snapshot of the volume will be available in the **`.snaps`** directory of every directory of the mounted volume.

Note

To access the snapshot you must first mount the volume.

For NFS mount refer Section 6.2.2.2.1, “Manually Mounting Volumes Using Gluster NFS” for more details. Following command is an example.

```
# mount -t nfs -o vers=3 server1:/test-vol /mnt/glusterfs
```

For FUSE mount refer Section 6.1.3.2, “Mounting Volumes Manually” for more details. Following command is an example.

```
# mount -t glusterfs server1:/test-vol /mnt/glusterfs
```

The **`.snaps`** directory is a virtual directory which will not be listed by either the**`ls`** command, or the **`ls -a`** option. The .snaps directory will contain every snapshot taken for that given volume as individual directories. Each of these snapshot entries will in turn contain the data of the particular directory the user is accessing from when the snapshot was taken.

To view or retrieve a file from a snapshot follow these steps:

1. Go to the folder where the file was present when the snapshot was taken. For example, if you had a test.txt file in the root directory of the mount that has to be recovered, then go to that directory.

   ```
   # cd /mnt/glusterfs
   ```

   Note

   Since every directory has a virtual **`.snaps`** directory, you can enter the **`.snaps`** directory from here. Since **`.snaps`** is a virtual directory, **`ls`** and **`ls -a`** command will not list the **`.snaps`** directory. For example:

   ```
   # ls -a
         ....Bob  John  test1.txt  test2.txt
   ```

2. Go to the **`.snaps`** folder

   ```
   # cd .snaps
   ```

3. Run the **`ls`** command to list all the snaps

   For example:

```
 # ls -p
 snapshot_Dec2014/    snapshot_Nov2014/    snapshot_Oct2014/
snapshot_Sept2014/
```

4. Go to the snapshot directory from where the file has to be retrieved.

   For example:

   ```
   cd snapshot_Nov2014
   ```

   ```
   # ls -p
       John/  test1.txt  test2.txt
   ```

5. Copy the file/directory to the desired location.

   ```
   # cp -p test2.txt  $HOME
   ```

8.13.3. Viewing and Retrieving Snapshots using CIFS for Windows Client

For every snapshot available for a volume, any user who has access to the volume will have a read-only view of the volume. You can recover the files through these read-only views of the volume from different point in time. Each snapshot of the volume will be available in the **`.snaps`** folder of every folder in the root of the CIFS share. The **`.snaps`** folder is a hidden folder which will be displayed only when the following option is set to **`ON`** on the volume using the following command:

```
# gluster volume set volname features.show-snapshot-directory on
```

After the option is set to **`ON`**, every Windows client can access the **`.snaps`** folder by following these steps:

1. In the **`Folder`** options, enable the **`Show hidden files, folders, and drives`** option.

2. Go to the root of the CIFS share to view the **`.snaps`** folder.

   > **Note**
   >
   > The **`.snaps`** folder is accessible only in the root of the CIFS share and not in any sub folders.

3. The list of snapshots are available in the **`.snaps`** folder. You can now access the required file and retrieve it.

You can also access snapshots on Windows using Samba. For more information see, Section 6.3.8, “Accessing Snapshots in Windows”.

8.14. Troubleshooting

Situation

Snapshot creation fails.

Step 1

Check if the bricks are thinly provisioned by following these steps:

- Execute the **mount** command and check the device name mounted on the brick path. For example:

  ```
  # mount
  /dev/mapper/snap_lvgrp-snap_lgvol on /rhgs/brick1 type xfs (rw)
  /dev/mapper/snap_lvgrp1-snap_lgvol1 on /rhgs/brick2 type xfs (rw)
  ```

- Run the following command to check if the device has a LV pool name.

  ```
  lvs device-name
  ```

  For example:

  ```
  #  lvs -o pool_lv /dev/mapper/snap_lvgrp-snap_lgvol
     Pool
     snap_thnpool
  ```

  If the **Pool** field is empty, then the brick is not thinly provisioned.

- Ensure that the brick is thinly provisioned, and retry the snapshot create command.

Step 2

Check if the bricks are down by following these steps:

- Execute the following command to check the status of the volume:

  ```
  # gluster volume status VOLNAME
  ```

- If any bricks are down, then start the bricks by executing the following command:

  ```
  # gluster volume start VOLNAME force
  ```

- To verify if the bricks are up, execute the following command:

  ```
  # gluster volume status VOLNAME
  ```

- Retry the snapshot create command.

Step 3

Check if the node is down by following these steps:

- Execute the following command to check the status of the nodes:

  ```
  # gluster volume status VOLNAME
  ```

- If a brick is not listed in the status, then execute the following command:

  ```
  # gluster pool list
  ```

- If the status of the node hosting the missing brick is **`Disconnected`**, then power-up the node.
- Retry the snapshot create command.

Step 4

Check if rebalance is in progress by following these steps:

- Execute the following command to check the rebalance status:

```
gluster volume rebalance VOLNAME status
```

- If rebalance is in progress, wait for it to finish.
- Retry the snapshot create command.

- ### Situation

Snapshot delete fails.

Step 1

Check if the server quorum is met by following these steps:

- Execute the following command to check the peer status:

```
# gluster pool list
```

- If nodes are down, and the cluster is not in quorum, then power up the nodes.
- To verify if the cluster is in quorum, execute the following command:

```
# gluster pool list
```

- Retry the snapshot delete command.

- ### Situation

Snapshot delete command fails on some node(s) during commit phase, leaving the system inconsistent.

Solution

- Identify the node(s) where the delete command failed. This information is available in the delete command's error output. For example:

```
# gluster snapshot delete snapshot1
Deleting snap will erase all the information about the snap. Do you
still want to continue? (y/n) y
snapshot delete: failed: Commit failed on 10.00.00.02. Please check
log file for details.
Snapshot command failed
```

- On the node where the delete command failed, bring down glusterd using the following command:

```
# service glusterd stop
```

Important

If **`glusterd`** crashes, there is no functionality impact to this crash as it occurs during the shutdown. For more information, see Section 24.3, “Resolving **`glusterd`** Crash”

- Delete that particular snaps repository in **`/var/lib/glusterd/snaps/`** from that node. For example:

```
# rm -rf /var/lib/glusterd/snaps/snapshot1
```

- Start glusterd on that node using the following command:

```
# service glusterd start.
```

- Repeat the 2nd, 3rd, and 4th steps on all the nodes where the commit failed as identified in the 1st step.

- Retry deleting the snapshot. For example:

```
# gluster snapshot delete snapshot1
```

- **Situation**

  Snapshot restore fails.

Step 1

Check if the server quorum is met by following these steps:

- Execute the following command to check the peer status:

```
# gluster pool list
```

- If nodes are down, and the cluster is not in quorum, then power up the nodes.

- To verify if the cluster is in quorum, execute the following command:

```
# gluster pool list
```

- Retry the snapshot restore command.

Step 2

Check if the volume is in **`Stop`** state by following these steps:

- Execute the following command to check the volume info:

```
# gluster volume info VOLNAME
```

- If the volume is in **`Started`** state, then stop the volume using the following command:

```
gluster volume stop VOLNAME
```

  - Retry the snapshot restore command.

- **Situation**

  Snapshot commands fail.

  Step 1

  Check if there is a mismatch in the operating versions by following these steps:

  - Open the following file and check for the operating version:

    ```
    /var/lib/glusterd/glusterd.info
    ```

    If the **`operating-version`** is lesser than 30000, then the snapshot commands are not supported in the version the cluster is operating on.

  - Upgrade all nodes in the cluster to Red Hat Gluster Storage 3.2 or higher.

  - Retry the snapshot command.

- **Situation**

  After rolling upgrade, snapshot feature does not work.

  Solution

  You must ensure to make the following changes on the cluster to enable snapshot:

  - Restart the volume using the following commands.

    ```
    # gluster volume stop VOLNAME
    # gluster volume start VOLNAME
    ```

  - Restart glusterd services on all nodes.

    ```
    # service glusterd restart
    ```

Chapter 9. Managing Directory Quotas

Quotas allow you to set limits on the disk space used by a directory. Storage administrators can control the disk space utilization at the directory and volume levels. This is particularly useful in cloud deployments to facilitate the use of utility billing models.

9.1. Enabling and Disabling Quotas

To limit disk usage, you need to enable quota usage on a volume by running the following command:

```
# gluster volume quota VOLNAME enable
```

This command only enables quota behavior on the volume; it does not set any default disk usage limits.

To disable quota behavior on a volume, including any set disk usage limits, run the following command:

```
# gluster volume quota VOLNAME disable
```

Important

When you disable quotas on Red Hat Gluster Storage 3.1.1 and earlier, all previously configured limits are removed from the volume by a cleanup process, **`quota-remove-xattr.sh`**. If you re-enable quotas while the cleanup process is still running, the extended attributes that enable quotas may be removed by the cleanup process. This has negative effects on quota accounting.

9.2. Before Setting a Quota on a Directory

There are several things you should keep in mind when you set a quota on a directory.

- When specifying a directory to limit with the **`gluster volume quota`** command, the directory's path is relative to the Red Hat Gluster Storage volume mount point, not the root directory of the server or client on which the volume is mounted. That is, if the Red Hat Gluster Storage volume is mounted at **`/mnt/glusterfs`** and you want to place a limit on the **`/mnt/glusterfs/dir`** directory, use **`/dir`** as the path when you run the **`gluster volume quota`** command, like so:

  ```
  # gluster volume quota VOLNAME limit-usage /dir hard_limit
  ```

- Ensure that at least one brick is available per replica set when you run the **`gluster volume quota`** command. A brick is available if a **`Y`** appears in the **`Online`** column of **`gluster volume status`** command output, like so:

  ```
  # gluster volume status VOLNAME
  Status of volume: VOLNAME
  Gluster process                          Port    Online   Pid
  ------------------------------------------------------------
  Brick arch:/export/rep1                  24010   Y       18474
  Brick arch:/export/rep2                  24011   Y       18479
  NFS Server on localhost                  38467   Y       18486
  Self-heal Daemon on localhost            N/A     Y       18491
  ```

9.3. Limiting Disk Usage

9.3.1. Setting Disk Usage Limits

If your system requires that a certain amount of space remains free in order to achieve a certain level of performance, you may need to limit the amount of space that Red Hat Gluster Storage consumes on a volume or directory.

Use the following command to limit the total allowed size of a directory, or the total amount of space to be consumed on a volume.

```
# gluster volume quota VOLNAME limit-usage path hard_limit
```

For example, to limit the size of the **/dir** directory on the **data** volume to 100 GB, run the following command:

```
# gluster volume quota data limit-usage /dir 100GB
```

This prevents the **/dir** directory and all files and directories underneath it from containing more than 100 GB of data cumulatively.

To limit the size of the entire **data** volume to 1 TB, set a 1 TB limit on the root directory of the volume, like so:

```
# gluster volume quota data limit-usage / 1TB
```

You can also set a percentage of the hard limit as a soft limit. Exceeding the soft limit for a directory logs warnings rather than preventing further disk usage. For example, to set a soft limit at 75% of your volume's hard limit of 1TB, run the following command.

```
# gluster volume quota data limit-usage / 1TB 75
```

By default, brick logs are found in **/var/log/glusterfs/bricks/*BRICKPATH*.log**.

The default soft limit is 80%. However, you can alter the default soft limit on a per-volume basis by using the **default-soft-limit** subcommand. For example, to set a default soft limit of 90% on the data volume, run the following command:

```
# gluster volume quota data default-soft-limit 90
```

Then verify that the new value is set with the following command:

```
# gluster volume quota VOLNAME list
```

Changing the default soft limit does not remove a soft limit set with the **limit-usage** subcommand.

9.3.2. Viewing Current Disk Usage Limits

You can view all of the limits currently set on a volume by running the following command:

```
# gluster volume quota VOLNAME list
```

For example, to view the quota limits set on *test-volume*:

```
# gluster volume quota test-volume list
Path         Hard-limit  Soft-limit   Used      Available
--------------------------------------------------------
/            50GB        75%          0Bytes    50.0GB
/dir         10GB        75%          0Bytes    10.0GB
/dir/dir2    20GB        90%          0Bytes    20.0GB
```

To view limit information for a particular directory, specify the directory path. Remember that the directory's path is relative to the Red Hat Gluster Storage volume mount point, not the root directory of the server or client on which the volume is mounted.

```
# gluster volume quota VOLNAME list /<directory_name>
```

For example, to view limits set on the */dir* directory of the *test-volume* volume:

```
# gluster volume quota test-volume list /dir
Path   Hard-limit   Soft-limit   Used   Available
-------------------------------------------------
/dir    10.0GB          75%       0Bytes  10.0GB
```

You can also list multiple directories to display disk limit information on each directory specified, like so:

```
# gluster volume quota VOLNAME list DIR1  DIR2
```

9.3.2.1. Viewing Quota Limit Information Using the df Utility

By default, the **df** utility does not take quota limits into account when reporting disk usage. This means that clients accessing directories see the total space available to the volume, rather than the total space allotted to their directory by quotas. You can configure a volume to display the hard quota limit as the total disk space instead by setting ***quota-deem-statfs*** parameter to **on**.

To set the ***quota-deem-statfs*** parameter to **on**, run the following command:

```
# gluster volume set VOLNAME quota-deem-statfs on
```

This configures **df** to to display the hard quota limit as the total disk space for a client.

The following example displays the disk usage as seen from a client when ***quota-deem-statfs*** is set to **off**:

```
# df -hT /home
Filesystem           Type             Size  Used Avail Use% Mounted on
server1:/test-volume fuse.glusterfs   400G   12G  389G   3% /home
```

The following example displays the disk usage as seen from a client when **quota-deem-statfs** is set to **on**:

```
# df -hT /home
Filesystem            Type             Size  Used Avail Use% Mounted on
server1:/test-volume  fuse.glusterfs   300G   12G  289G   4% /home
```

9.3.3. Setting Quota Check Frequency (Timeouts)

You can configure how frequently Red Hat Gluster Storage checks disk usage against the disk usage limit by specifying soft and hard timeouts.

The **soft-timeout** parameter specifies how often Red Hat Gluster Storage checks space usage when usage has, so far, been below the soft limit set on the directory or volume. The default soft timeout frequency is every **60** seconds.

To specify a different soft timeout, run the following command:

```
# gluster volume quota VOLNAME soft-timeout seconds
```

The **hard-timeout** parameter specifies how often Red Hat Gluster Storage checks space usage when usage is greater than the soft limit set on the directory or volume. The default hard timeout frequency is every **5** seconds.

To specify a different hard timeout, run the following command:

```
# gluster volume quota VOLNAME hard-timeout seconds
```

Important

Ensure that you take system and application workload into account when you set soft and hard timeouts, as the margin of error for disk usage is proportional to system workload.

9.3.4. Setting Logging Frequency (Alert Time)

The **alert-time** parameter configures how frequently usage information is logged after the soft limit has been reached. You can configure **alert-time** with the following command:

```
# gluster volume quota VOLNAME alert-time time
```

By default, alert time is 1 week (**1w**).

The *time* parameter in the command can be used with one of the following formats:

Table 9.1.

Unit of time	Format 1	Format 2
Second(s)	*[integer]*s	*[integer]*sec
Minute(s)	*[integer]*m	*[integer]*min
Hour(s)	*[integer]*h	*[integer]*hr
Day(s)	*[integer]*d	*[integer]*days
Week(s)	*[integer]*w	*[integer]*wk

The *[integer]* is the number of units of time that need to be provided. Any one of the format for any unit of time can be used. For example:

The following command sets the logging frequency for volume named test-vol to every 10 minutes.

```
# gluster volume quota test-vol alert-time 10m
```

Whereas, the following command will set the logging frequency for volume named test-vol to every 10 days.

```
# gluster volume quota test-vol alert-time 10days
```

9.3.5. Removing Disk Usage Limits

If you don't need to limit disk usage, you can remove the usage limits on a directory by running the following command:

```
# gluster volume quota VOLNAME remove DIR
```

For example, to remove the disk limit usage on */data* directory of *test-volume*:

```
# gluster volume quota test-volume remove /data
  volume quota : success
```

To remove a volume-wide quota, run the following command:

```
# gluster vol quota VOLNAME remove /
```

This does not remove limits recursively; it only impacts a volume-wide limit.

Chapter 10. Managing Geo-replication

This section introduces geo-replication, illustrates the various deployment scenarios, and explains how to configure geo-replication and mirroring.

10.1. About Geo-replication

Geo-replication provides a distributed, continuous, asynchronous, and incremental replication service from one site to another over Local Area Networks (LANs), Wide Area Networks (WANs), and the Internet.

Geo-replication uses a master–slave model, where replication and mirroring occurs between the following partners:

- Master – the primary Red Hat Gluster Storage volume.
- Slave – a secondary Red Hat Gluster Storage volume. A slave volume can be a volume on a remote host, such as **`remote-host::volname`**.

10.2. Replicated Volumes vs Geo-replication

The following table lists the differences between replicated volumes and geo-replication:

Replicated Volumes	Geo-replication
Works between all bricks in a replica set, so that changes are synced in both directions.	Works only from the primary (master) volume to the secondary (slave) volume.
Mirrors data across bricks within one trusted storage pool.	Mirrors data across geographically distributed trusted storage pools.
Provides high-availability.	Provides data back-up for disaster recovery.
Synchronous replication: each and every file operation is applied to all the bricks.	Asynchronous replication: checks for changes in files periodically, and syncs them on detecting differences.

10.3. Preparing to Deploy Geo-replication

This section provides an overview of geo-replication deployment scenarios, lists prerequisites, and describes how to setup the environment for geo-replication session.

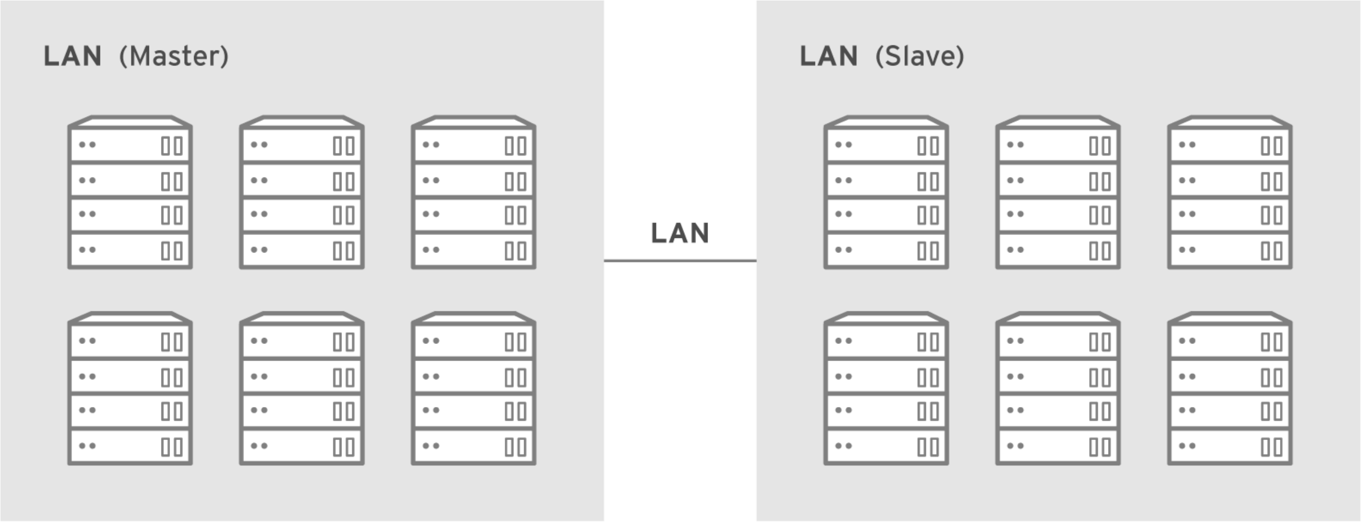
- Section 10.3.1, “Exploring Geo-replication Deployment Scenarios”
- Section 10.3.2, “Geo-replication Deployment Overview”
- Section 10.3.3, “Prerequisites”
- Section 10.3.4.2, “Setting Up your Environment for a Secure Geo-replication Slave”
- Section 10.3.4.1, “Setting Up your Environment for Geo-replication Session”
- Section 10.3.5, “Configuring a Meta-Volume”

10.3.1. Exploring Geo-replication Deployment Scenarios

Geo-replication provides an incremental replication service over Local Area Networks (LANs), Wide Area Network (WANs), and the Internet. This section illustrates the most common deployment scenarios for geo-replication, including the following:

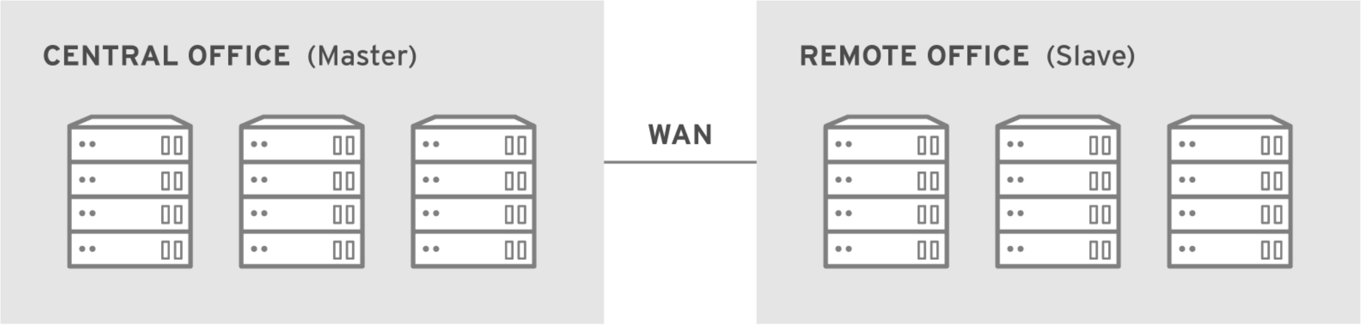
- Geo-replication over LAN
- Geo-replication over WAN
- Geo-replication over the Internet
- Multi-site cascading geo-replication

Geo-replication over LAN



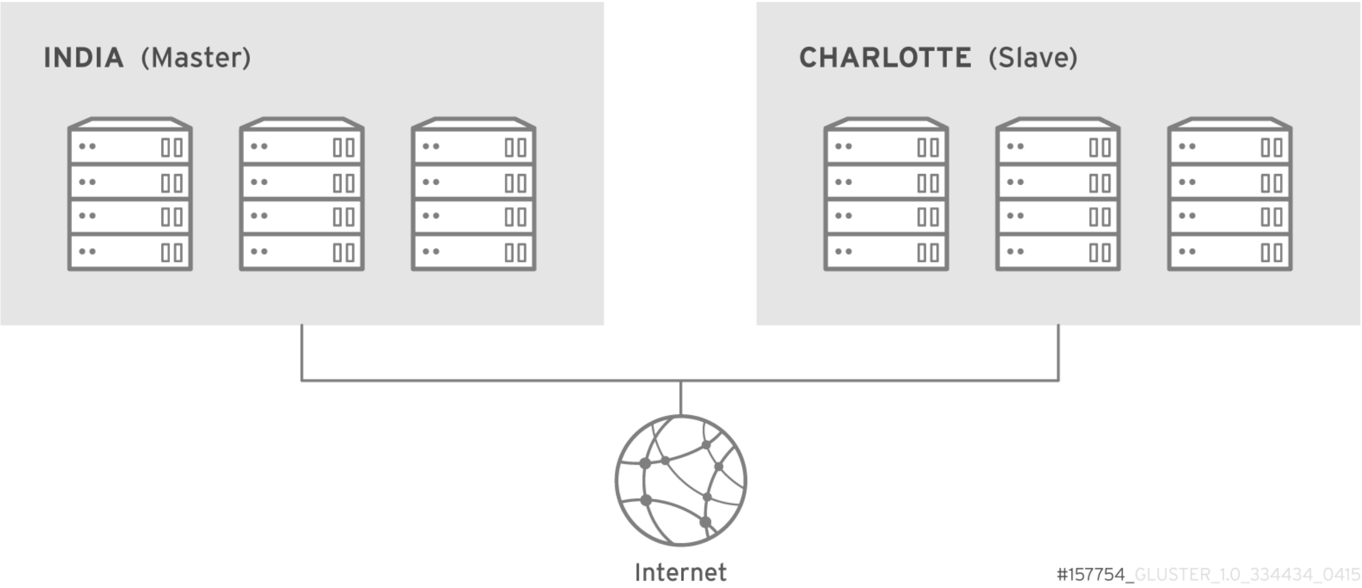
#157752_GLUSTER_1.0_334434_0415

Geo-replication over WAN

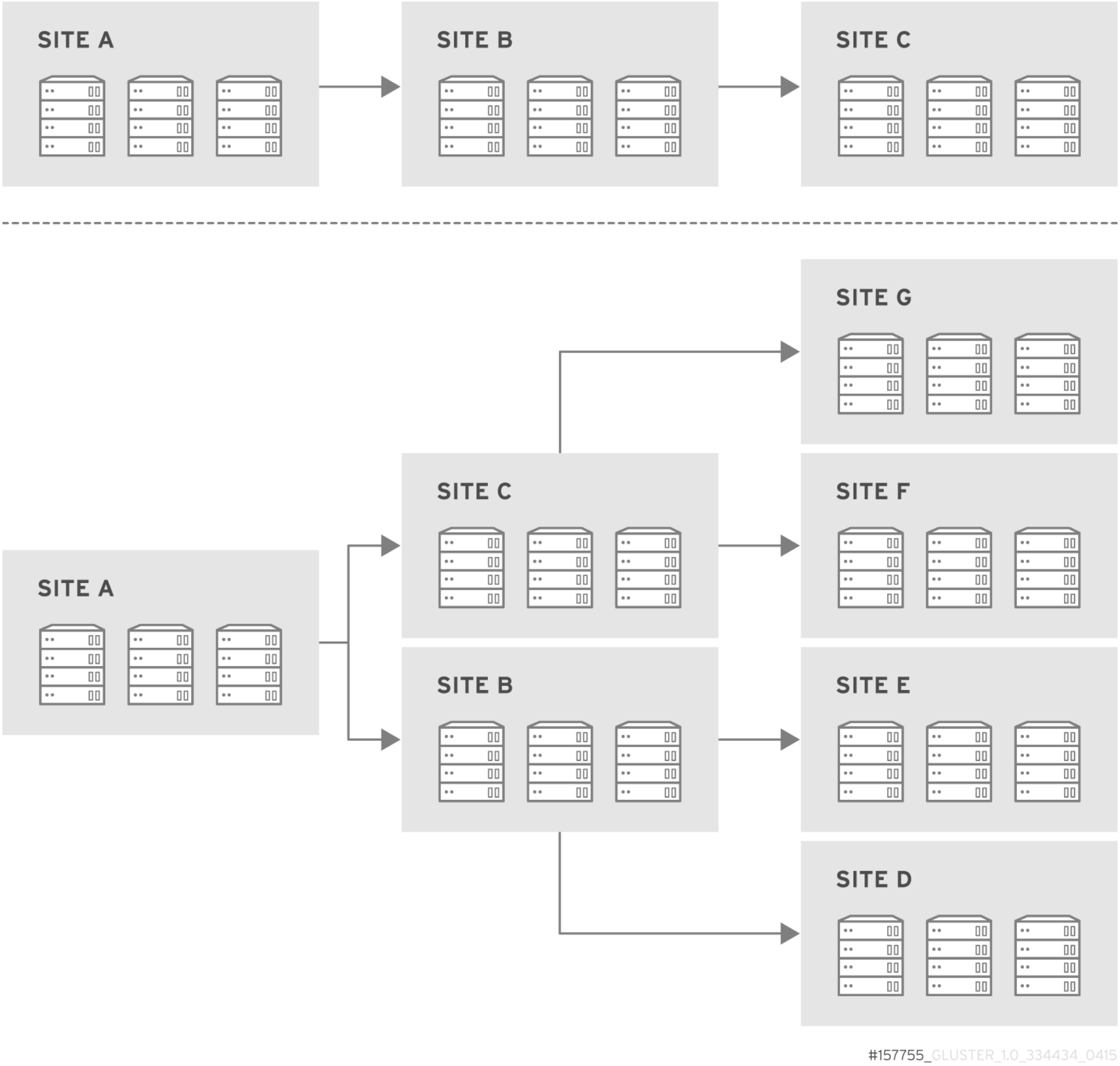


#157753_GLUSTER_1.0_334434_0415

Geo-replication over Internet



Multi-site cascading Geo-replication



10.3.2. Geo-replication Deployment Overview

Deploying geo-replication involves the following steps:

1. Verify that your environment matches the minimum system requirements. See Section 10.3.3, “Prerequisites”.
2. Determine the appropriate deployment scenario. See Section 10.3.1, “Exploring Geo-replication Deployment Scenarios”.
3. Start geo-replication on the master and slave systems. See Section 10.4, “Starting Geo-replication”.

10.3.3. Prerequisites

The following are prerequisites for deploying geo-replication:

Note that these prerequisites only need to be carried out once from one cluster to another cluster, so if you are syncing multiple volumes from the same master cluster to the same slave cluster, you need only perform these prerequisites once.

- The master and slave volumes must use the same version of Red Hat Gluster Storage.
- Nodes in the slave volume must not be part of the master volume. Two separate trusted storage pools are required.
- Disable the **performance.quick-read** option in the slave volume using the following command:

  ```
  [slave ~]# gluster volume set slavevol performance.quick-read off
  ```

- Time must be synchronized between all master and slave nodes before geo-replication is configured. Red Hat recommends setting up a network time protocol service to keep time synchronized between bricks and servers, and avoid out-of-time synchronization errors.

  See Network Time Protocol Setup for more information.

- Add the required port for geo-replication from the ports listed in the Table 3.2, “TCP Port Numbers used for Geo-replication”.
- Key-based SSH authentication without a password is required between one node of the master volume (the node from which the **geo-replication create** command will be executed), and one node of the slave volume (the node whose IP/hostname will be mentioned in the slave name when running the **geo-replication create** command).

  Create the public and private keys using **ssh-keygen** (without passphrase) on the master node:

  ```
  # ssh-keygen
  ```

  Copy the public key to the slave node using the following command:

  ```
  # ssh-copy-id -i identity_file root@slave_node_IPaddress/Hostname
  ```

  If you are setting up a non-root geo-replicaton session, then copy the public key to the respective **user** location.

Note

- Key-based SSH authentication without a password is only required from the master node to the slave node; the slave node does not need this level of access.

- **ssh-copy-id** command does not work if **ssh authorized_keys** file is configured in the custom location. You must copy the contents of **.ssh/id_rsa.pub** file from the Master and paste it to authorized_keys file in the custom location on the Slave node.

Gsyncd also requires key-based SSH authentication without a password between every node in the master cluster to every node in the slave cluster. The **gluster system:: execute gsec_create** command creates **secret-pem** files on all the nodes in the master, and is used to implement the SSH authentication connection. The **push-pem** option in the **geo-replication create** command pushes these keys to all slave nodes.

For more information on the **gluster system::execute gsec_create** and **push-pem** commands, see Section 10.3.4.1, “Setting Up your Environment for Geo-replication Session”.

10.3.4. Setting Up your Environment

You can set up your environment for a geo-replication session in the following ways:

- Section 10.3.4.1, “Setting Up your Environment for Geo-replication Session” - In this method, the slave mount is owned by the root user.

- Section 10.3.4.2, “Setting Up your Environment for a Secure Geo-replication Slave” - This method is more secure as the slave mount is owned by a normal user.

10.3.4.1. Setting Up your Environment for Geo-replication Session

Creating Geo-replication Sessions

1. To create a common **pem pub** file, run the following command on the master node where the key-based SSH authentication connection is configured:

   ```
   # gluster system:: execute gsec_create
   ```

2. Create the geo-replication session using the following command. The **push-pem** option is needed to perform the necessary **pem-file** setup on the slave nodes.

   ```
   # gluster volume geo-replication MASTER_VOL SLAVE_HOST::SLAVE_VOL
   create push-pem [force]
   ```

   For example:

   ```
   # gluster volume geo-replication Volume1 storage.backup.com::slave-vol
   create push-pem
   ```

> **Note**
>
> - There must be key-based SSH authentication access between the node from which this command is run, and the slave host specified in the above command. This command performs the slave verification, which includes checking for a valid slave URL, valid slave volume, and available space on the slave. If the verification fails, you can use the **`force`** option which will ignore the failed verification and create a geo-replication session.
> - The slave volume is in read-only mode by default. However, in case of a failover-failback situation, the original master is made read-only by default as the session is from the original slave to the original master.

3. Enable shared storage for master and slave volumes:

   ```
   # gluster volume set all cluster.enable-shared-storage enable
   ```

   For more information on shared storage, see Section 11.12, “Setting up Shared Storage Volume”.

4. Configure the meta-volume for geo-replication:

   ```
   # gluster volume geo-replication MASTER_VOL SLAVE_HOST::SLAVE_VOL
   config use_meta_volume true
   ```

   For example:

   ```
   # gluster volume geo-replication Volume1 storage.backup.com::slave-vol
   config use_meta_volume true
   ```

   For more information on configuring meta-volume, see Section 10.3.5, “Configuring a Meta-Volume”.

5. Start the geo-replication by running the following command on the master node:

   For example,

   ```
   # gluster volume geo-replication MASTER_VOL SLAVE_HOST::SLAVE_VOL
   start [force]
   ```

6. Verify the status of the created session by running the following command:

   ```
   # gluster volume geo-replication MASTER_VOL SLAVE_HOST::SLAVE_VOL
   status
   ```

10.3.4.2. Setting Up your Environment for a Secure Geo-replication Slave

Geo-replication supports access to Red Hat Gluster Storage slaves through SSH using an unprivileged account (user account with non-zero UID). This method is more secure and it reduces the master's capabilities over slave to the minimum. This feature relies on **`mountbroker`**, an internal service of glusterd which manages the mounts for unprivileged slave accounts. You must perform additional steps to configure glusterd with the appropriate **`mountbroker's`** access control directives. The following example demonstrates this process:

Perform the following steps on all the Slave nodes to setup an auxiliary glusterFS mount for the unprivileged account:

1. In all the slave nodes, create a new group. For example, **`geogroup`**.

   > **Note**
   >
   > You must not use multiple groups for the **`mountbroker`** setup. You can create multiple user accounts but the group should be same for all the non-root users.

2. In all the slave nodes, create a unprivileged account. For example, **`geoaccount`**. Add **`geoaccount`** as a member of **`geogroup`** group.

3. On any one of the Slave nodes, run the following command to set up mountbroker root directory and group.

   ```
   # gluster-mountbroker setup <MOUNT ROOT> <GROUP>
   ```

   For example,

   ```
   # gluster-mountbroker setup /var/mountbroker-root geogroup
   ```

4. On any one of the Slave nodes, run the following commands to add volume and user to the mountbroker service.

   ```
   # gluster-mountbroker add <VOLUME> <USER>
   ```

   For example,

   ```
   # gluster-mountbroker add slavevol geoaccount
   ```

5. Check the status of the setup by running the following command:

   ```
   # gluster-mountbroker status

        NODE    NODE STATUS                  MOUNT ROOT         GROUP
   USERS
   ------------------------------------------------------------------
   --------------------- localhost             UP   /var/mountbroker-
   root(OK)  geogroup(OK)  geoaccount(slavevol)
       node2             UP   /var/mountbroker-root(OK)  geogroup(OK)
   geoaccount(slavevol)
   ```

   The output displays the mountbroker status for every peer node in the slave cluster.

6. Restart **`glusterd`** service on all the Slave nodes.

   ```
   # service glusterd restart
   ```

   After you setup an auxiliary glusterFS mount for the unprivileged account on all the Slave nodes, perform the following steps to setup a non-root geo-replication session.:

7. Setup key-based SSH authentication from one of the master nodes to the **`user`** on one of the slave nodes.

For example, to setup key-based SSH authentication to the user *geoaccount*.

```
# ssh-keygen
# ssh-copy-id -i identity_file
geoaccount@slave_node_IPaddress/Hostname
```

8. Create a common pem pub file by running the following command on the master nodes, where the key-based SSH authentication connection is configured to the **`user`** on the slave nodes:

```
# gluster system:: execute gsec_create
```

9. Create a geo-replication relationship between the master and the slave to the **`user`** by running the following command on the master node:

   For example,

```
# gluster volume geo-replication MASTERVOL
geoaccount@SLAVENODE::slavevol create push-pem
```

   If you have multiple slave volumes and/or multiple accounts, create a geo-replication session with that particular user and volume.

   For example,

```
# gluster volume geo-replication MASTERVOL
geoaccount2@SLAVENODE::slavevol2 create push-pem
```

10. On the slave node, which is used to create relationship, run **`/usr/libexec/glusterfs/set_geo_rep_pem_keys.sh`** as a root with user name, master volume name, and slave volume names as the arguments.

    For example,

```
# /usr/libexec/glusterfs/set_geo_rep_pem_keys.sh geoaccount MASTERVOL
SLAVEVOL_NAME
```

11. Configure the meta-volume for geo-replication:

```
# gluster volume geo-replication MASTER_VOL SLAVE_HOST::SLAVE_VOL
config use_meta_volume true
```

    For example:

```
# gluster volume geo-replication Volume1 storage.backup.com::slave-vol
config use_meta_volume true
```

    For more information on configuring meta-volume, see Section 10.3.5, “Configuring a Meta-Volume”.

12. Start the geo-replication with slave user by running the following command on the master node:

    For example,

```
# gluster volume geo-replication MASTERVOL
geoaccount@SLAVENODE::slavevol start
```

13. Verify the status of geo-replication session by running the following command on the master node:

    ```
    # gluster volume geo-replication MASTERVOL
    geoaccount@SLAVENODE::slavevol status
    ```

Deleting a mountbroker geo-replication options after deleting session

After mountbroker geo-replicaton session is deleted, use the following command to remove volumes per mountbroker user.

```
# gluster-mountbroker remove [--volume volume] [--user user]
```

For example,

```
# gluster-mountbroker remove --volume slavevol --user geoaccount
# gluster-mountbroker remove --user geoaccount
# gluster-mountbroker remove --volume slavevol
```

If the volume to be removed is the last one for the mountbroker user, the user is also removed.

Important

If you have a secured geo-replication setup, you must ensure to prefix the unprivileged user account to the slave volume in the command. For example, to execute a geo-replication status command, run the following:

```
# gluster volume geo-replication MASTERVOL
geoaccount@SLAVENODE::slavevol status
```

In this command, **geoaccount** is the name of the unprivileged user account.

10.3.5. Configuring a Meta-Volume

For effective handling of node fail-overs in Master volume, geo-replication requires a shared storage to be available across all nodes of the cluster. Hence, you must ensure that a gluster volume named **gluster_shared_storage** is created in the cluster, and is mounted at **/var/run/gluster/shared_storage** on all the nodes in the cluster. For more information on setting up shared storage volume, see Section 11.12, “Setting up Shared Storage Volume”.

- Configure the meta-volume for geo-replication:

  ```
  # gluster volume geo-replication MASTER_VOL SLAVE_HOST::SLAVE_VOL config
  use_meta_volume true
  ```

  For example:

  ```
  # gluster volume geo-replication Volume1 storage.backup.com::slave-vol
  config use_meta_volume true
  ```

10.4. Starting Geo-replication

10.4. Starting Geo-replication

This section describes how to and start geo-replication in your storage environment, and verify that it is functioning correctly.

- Section 10.4.1, “Starting a Geo-replication Session”
- Section 10.4.2, “Verifying a Successful Geo-replication Deployment”
- Section 10.4.3, “Displaying Geo-replication Status Information”
- Section 10.4.4, “Configuring a Geo-replication Session”
- Section 10.4.5, “Stopping a Geo-replication Session”
- Section 10.4.6, “Deleting a Geo-replication Session”

10.4.1. Starting a Geo-replication Session

> **Important**
>
> You must create the geo-replication session before starting geo-replication. For more information, see Section 10.3.4.1, “Setting Up your Environment for Geo-replication Session”.

To start geo-replication, use one of the following commands:

- To start the geo-replication session between the hosts:

  ```
  # gluster volume geo-replication MASTER_VOL SLAVE_HOST::SLAVE_VOL start
  ```

  For example:

  ```
  # gluster volume geo-replication Volume1 storage.backup.com::slave-vol
  start
  Starting geo-replication session between Volume1 &
  storage.backup.com::slave-vol has been successful
  ```

  This command will start distributed geo-replication on all the nodes that are part of the master volume. If a node that is part of the master volume is down, the command will still be successful. In a replica pair, the geo-replication session will be active on any of the replica nodes, but remain passive on the others.

  After executing the command, it may take a few minutes for the session to initialize and become stable.

  > **Note**
  >
  > If you attempt to create a geo-replication session and the slave already has data, the following error message will be displayed:
  >
  > ```
  > slave-node::slave is not empty. Please delete existing files in
  > slave-node::slave and retry, or use force to continue without
  > deleting the existing files. geo-replication command failed
  > ```

- To start the geo-replication session *forcefully* between the hosts:

  ```
  # gluster volume geo-replication MASTER_VOL SLAVE_HOST::SLAVE_VOL start
  force
  ```

  For example:

  ```
  # gluster volume geo-replication Volume1 storage.backup.com::slave-vol
  start force
  Starting geo-replication session between Volume1 &
  storage.backup.com::slave-vol has been successful
  ```

  This command will force start geo-replication sessions on the nodes that are part of the master volume. If it is unable to successfully start the geo-replication session on any node which is online and part of the master volume, the command will still start the geo-replication sessions on as many nodes as it can. This command can also be used to re-start geo-replication sessions on the nodes where the session has died, or has not started.

10.4.2. Verifying a Successful Geo-replication Deployment

You can use the **status** command to verify the status of geo-replication in your environment:

```
# gluster volume geo-replication MASTER_VOL SLAVE_HOST::SLAVE_VOL status
```

For example:

```
# gluster volume geo-replication Volume1 storage.backup.com::slave-vol
status
```

10.4.3. Displaying Geo-replication Status Information

The **status** command can be used to display information about a specific geo-replication master session, master-slave session, or all geo-replication sessions. The status output provides both node and brick level information.

- To display information about all geo-replication sessions, use the following command:

  ```
  # gluster volume geo-replication status [detail]
  ```

- To display information on all geo-replication sessions from a particular master volume, use the following command:

  ```
  # gluster volume geo-replication MASTER_VOL status [detail]
  ```

- To display information of a particular master-slave session, use the following command:

  ```
  # gluster volume geo-replication MASTER_VOL SLAVE_HOST::SLAVE_VOL status
  [detail]
  ```

Important

There will be a mismatch between the outputs of the `df` command (including `-h` and `-k`) and inode of the master and slave volumes when the data is in full sync. This is due to the extra inode and size consumption by the `changelog` journaling data, which keeps track of the changes done on the file system on the `master` volume. Instead of running the `df` command to verify the status of synchronization, use `# gluster volume geo-replication MASTER_VOL SLAVE_HOST::SLAVE_VOL status detail` instead.

- The geo-replication status command output provides the following information:
  - **Master Node**: Master node and Hostname as listed in the `gluster volume info` command output
  - **Master Vol**: Master volume name
  - **Master Brick**: The path of the brick
  - **Slave User**: Slave user name
  - **Slave**: Slave volume name
  - **Slave Node**: IP address/hostname of the slave node to which master worker is connected to.
  - **Status**: The status of the geo-replication worker can be one of the following:
    - **Initializing**: This is the initial phase of the Geo-replication session; it remains in this state for a minute in order to make sure no abnormalities are present.
    - **Created**: The geo-replication session is created, but not started.
    - **Active**: The **gsync** daemon in this node is active and syncing the data.
    - **Passive**: A replica pair of the active node. The data synchronization is handled by the active node. Hence, this node does not sync any data.
    - **Faulty**: The geo-replication session has experienced a problem, and the issue needs to be investigated further. For more information, see Section 10.11, “Troubleshooting Geo-replication” section.
    - **Stopped**: The geo-replication session has stopped, but has not been deleted.
  - **Crawl Status**: Crawl status can be one of the following:
    - **Changelog Crawl:** The `changelog` translator has produced the changelog and that is being consumed by `gsyncd` daemon to sync data.
    - **Hybrid Crawl**: The `gsyncd` daemon is crawling the glusterFS file system and generating pseudo changelog to sync data.
    - **History Crawl**: The `gsyncd` daemon consumes the history changelogs produced by the changelog translator to sync data.
  - **Last Synced**: The last synced time.
  - **Entry**: The number of pending entry (CREATE, MKDIR, RENAME, UNLINK etc) operations per session.
  - **Data**: The number of `Data` operations pending per session.

- **Meta**: The number of **Meta** operations pending per session.
- **Failures**: The number of failures. If the failure count is more than zero, view the log files for errors in the Master bricks.
- **Checkpoint Time**: Displays the date and time of the checkpoint, if set. Otherwise, it displays as N/A.
- **Checkpoint Completed**: Displays the status of the checkpoint.
- **Checkpoint Completion Time**: Displays the completion time if Checkpoint is completed. Otherwise, it displays as N/A.

10.4.4. Configuring a Geo-replication Session

To configure a geo-replication session, use the following command:

```
# gluster volume geo-replication MASTER_VOL SLAVE_HOST::SLAVE_VOL config
[Name] [Value]
```

For example:

```
# gluster volume geo-replication Volume1 storage.backup.com::slave-vol
config use_tarssh true
```

For example, to view the list of all option/value pairs:

```
# gluster volume geo-replication Volume1 storage.backup.com::slave-vol
config
```

To delete a setting for a geo-replication config option, prefix the option with **!** (exclamation mark). For example, to reset **log-level** to the default value:

```
# gluster volume geo-replication Volume1 storage.backup.com::slave-vol
config '!log-level'
```

Warning

You must ensure to perform these configuration changes when all the peers in cluster are in **Connected** (online) state. If you change the configuration when any of the peer is down, the geo-replication cluster would be in inconsistent state when the node comes back online.

Configurable Options

The following table provides an overview of the configurable options for a geo-replication setting:

Option	Description
gluster-log-file *LOGFILE*	The path to the geo-replication glusterfs log file.
gluster-log-level *LOGFILELEVEL*	The log level for glusterfs processes.
log-file *LOGFILE*	The path to the geo-replication log file.
log-level *LOGFILELEVEL*	The log level for geo-replication.

Option	Description
changelog-log-level *LOGFILELEVEL*	The log level for the changelog. The default log level is set to INFO.
changelog-batch-size *SIZEINBYTES*	The total size for the changelog in a batch. The default size is set to 727040 bytes.
ssh-command *COMMAND*	The SSH command to connect to the remote machine (the default is **`SSH`**).
rsync-command *COMMAND*	The rsync command to use for synchronizing the files (the default is **`rsync`**).
use-tarssh *[true | false]*	The use-tarssh command allows tar over Secure Shell protocol. Use this option to handle workloads of files that have not undergone edits.
volume_id=*UID*	The command to delete the existing master UID for the intermediate/slave node.
timeout *SECONDS*	The timeout period in seconds.
sync-jobs *N*	The number of sync-jobs represents the maximum number of syncer threads (rsync processes or tar over ssh processes for syncing) inside each worker. The number of workers is always equal to the number of bricks in the Master volume. For example, a distributed-replicated volume of (3 x 2) with sync-jobs configured at 3 results in 9 total sync-jobs (aka threads) across all nodes/servers. **`Active and Passive Workers`**: The number of active workers is based on the volume configuration. In case of a distribute volume, all bricks (workers) will be active and participate in syncing. In case of replicate or dispersed volume, one worker from each replicate/disperse group (subvolume) will be active and participate in syncing. This is to avoid duplicate syncing from other bricks. The remaining workers in each replicate/disperse group (subvolume) will be passive. In case the active worker goes down, one of the passive worker from the same replicate/disperse group will become an active worker.
ignore-deletes	If this option is set to **`1`**, a file deleted on the master will not trigger a delete operation on the slave. As a result, the slave will remain as a superset of the master and can be used to recover the master in the event of a crash and/or accidental delete.
checkpoint *[LABEL|now]*	Sets a checkpoint with the given option *LABEL*. If the option is set as **`now`**, then the current time will be used as the label.

Option	Description
sync-acls *[true | false]*	Syncs acls to the Slave cluster. By default, this option is enabled. Note Geo-replication can sync acls only with **rsync** as the sync engine and not with **tarssh** as the sync engine.
sync-xattrs *[true | false]*	Syncs extended attributes to the Slave cluster. By default, this option is enabled. Note Geo-replication can sync extended attributes only with **rsync** as the sync engine and not with **tarssh** as the sync engine.
log-rsync-performance *[true | false]*	If this option is set to **enable**, geo-replication starts recording the rsync performance in log files. By default, this option is disabled.
rsync-options	Additional options to rsync. For example, you can limit the rsync bandwidth usage "--bwlimit=<value>".
use-meta-volume *[true | false]*	Set this option to **enable**, to use meta volume in Geo-replication. By default, this option is disabled. Note For more information on meta-volume, see Section 10.3.5, “Configuring a Meta-Volume”.
meta-volume-mnt *PATH*	The path of the meta volume mount point.
gfid-conflict-resolution *[true | false]*	Auto GFID conflict resolution feature provides an ability to automatically detect and fix the GFID conflicts between master and slave. This configuration option provides an ability to enable or disable this feature. By default, this option is **true**.

10.4.4.1. Geo-replication Checkpoints

10.4.4.1.1. About Geo-replication Checkpoints

Geo-replication data synchronization is an asynchronous process, so changes made on the master may take time to be replicated to the slaves. Data replication to a slave may also be interrupted by various issues, such network outages.

Red Hat Gluster Storage provides the ability to set geo-replication checkpoints. By setting a checkpoint, synchronization information is available on whether the data that was on the master at that point in time has been replicated to the slaves.

10.4.4.1.2. Configuring and Viewing Geo-replication Checkpoint Information

- To set a checkpoint on a geo-replication session, use the following command:

  ```
  # gluster volume geo-replication MASTER_VOL SLAVE_HOST::SLAVE_VOL config
  checkpoint [now|LABEL]
  ```

  For example, to set checkpoint between **Volume1** and **storage.backup.com:/data/remote_dir**:

  ```
  # gluster volume geo-replication Volume1 storage.backup.com::slave-vol
  config checkpoint now
  geo-replication config updated successfully
  ```

  The label for a checkpoint can be set as the current time using **now**, or a particular label can be specified, as shown below:

  ```
  # gluster volume geo-replication Volume1 storage.backup.com::slave-vol
  config checkpoint NEW_ACCOUNTS_CREATED
  geo-replication config updated successfully.
  ```

- To display the status of a checkpoint for a geo-replication session, use the following command:

  ```
  # gluster volume geo-replication MASTER_VOL SLAVE_HOST::SLAVE_VOL status
  detail
  ```

- To delete checkpoints for a geo-replication session, use the following command:

  ```
  # gluster volume geo-replication MASTER_VOL SLAVE_HOST::SLAVE_VOL config
  '!checkpoint'
  ```

  For example, to delete the checkpoint set between **Volume1** and **storage.backup.com::slave-vol**:

  ```
  # gluster volume geo-replication Volume1 storage.backup.com::slave-vol
  config '!checkpoint'
  geo-replication config updated successfully
  ```

10.4.5. Stopping a Geo-replication Session

To stop a geo-replication session, use one of the following commands:

- To stop a geo-replication session between the hosts:

  ```
  # gluster volume geo-replication MASTER_VOL SLAVE_HOST::SLAVE_VOL stop
  ```

  For example:

```
# gluster volume geo-replication Volume1 storage.backup.com::slave-vol
stop
Stopping geo-replication session between Volume1 &
storage.backup.com::slave-vol has been successful
```

> **Note**
>
> The **stop** command will fail if:
>
> - any node that is a part of the volume is offline.
> - if it is unable to stop the geo-replication session on any particular node.
> - if the geo-replication session between the master and slave is not active.

- To stop a geo-replication session *forcefully* between the hosts:

```
# gluster volume geo-replication MASTER_VOL SLAVE_HOST::SLAVE_VOL stop
force
```

For example:

```
# gluster volume geo-replication Volume1 storage.backup.com::slave-vol
stop force
Stopping geo-replication session between Volume1 &
storage.backup.com::slave-vol has been successful
```

Using **force** will stop the geo-replication session between the master and slave even if any node that is a part of the volume is offline. If it is unable to stop the geo-replication session on any particular node, the command will still stop the geo-replication sessions on as many nodes as it can. Using **force** will also stop inactive geo-replication sessions.

10.4.6. Deleting a Geo-replication Session

> **Important**
>
> You must first stop a geo-replication session before it can be deleted. For more information, see Section 10.4.5, “Stopping a Geo-replication Session”.

To delete a geo-replication session, use the following command:

```
# gluster volume geo-replication MASTER_VOL SLAVE_HOST::SLAVE_VOL delete
[reset-sync-time]
```

reset-sync-time: The geo-replication delete command retains the information about the last synchronized time. Due to this, if the same geo-replication session is recreated, then the synchronization will continue from the time where it was left before deleting the session. For the geo-replication session to not maintain any details about the deleted session, use the **reset-sync-time** option with the delete command. Now, when the session is recreated, it starts synchronization from the beginning just like a new session.

For example:

```
# gluster volume geo-replication Volume1 storage.backup.com::slave-vol
delete
geo-replication command executed successfully
```

Note

The **delete** command will fail if:

- any node that is a part of the volume is offline.
- if it is unable to delete the geo-replication session on any particular node.
- if the geo-replication session between the master and slave is still active.

Important

The SSH keys will not removed from the master and slave nodes when the geo-replication session is deleted. You can manually remove the **pem** files which contain the SSH keys from the **/var/lib/glusterd/geo-replication/** directory.

10.5. Starting Geo-replication on a Newly Added Brick, Node, or Volume

10.5.1. Starting Geo-replication for a New Brick or New Node

If a geo-replication session is running, and a new node is added to the trusted storage pool or a brick is added to the volume from a newly added node in the trusted storage pool, then you must perform the following steps to start the geo-replication daemon on the new node:

1. Run the following command on the master node where key-based SSH authentication connection is configured, in order to create a common **pem pub** file.

   ```
   # gluster system:: execute gsec_create
   ```

2. Create the geo-replication session using the following command. The **push-pem** and **force** options are required to perform the necessary **pem-file** setup on the slave nodes.

   ```
   # gluster volume geo-replication MASTER_VOL SLAVE_HOST::SLAVE_VOL
   create push-pem force
   ```

   For example:

   ```
   # gluster volume geo-replication Volume1 storage.backup.com::slave-vol
   create push-pem force
   ```

Note

There must be key-based SSH authentication access between the node from which this command is run, and the slave host specified in the above command. This command performs the slave verification, which includes checking for a valid slave URL, valid slave volume, and available space on the slave.

3. After successfully setting up the shared storage volume, when a new node is added to the cluster, the shared storage is not mounted automatically on this node. Neither is the **/etc/fstab** entry added for the shared storage on this node. To make use of shared storage on this node, execute the following commands:

   ```
   # mount -t glusterfs <local node's ip>:gluster_shared_storage
   /var/run/gluster/shared_storage
   # cp /etc/fstab /var/run/gluster/fstab.tmp
   # echo "<local node's ip>:/gluster_shared_storage
   /var/run/gluster/shared_storage/ glusterfs defaults 0 0" >> /etc/fstab
   ```

   For more information on setting up shared storage volume, see Section 11.12, “Setting up Shared Storage Volume”.

4. Configure the meta-volume for geo-replication:

   ```
   # gluster volume geo-replication MASTER_VOL SLAVE_HOST::SLAVE_VOL
   config use_meta_volume true
   ```

   For example:

   ```
   # gluster volume geo-replication Volume1 storage.backup.com::slave-vol
   config use_meta_volume true
   ```

   For more information on configuring meta-volume, see Section 10.3.5, “Configuring a Meta-Volume”.

5. If a node is added at slave, stop the geo-replication session using the following command:

   ```
   # gluster volume geo-replication MASTER_VOL SLAVE_HOST::SLAVE_VOL stop
   ```

6. Start the geo-replication session between the slave and master forcefully, using the following command:

   ```
   # gluster volume geo-replication MASTER_VOL SLAVE_HOST::SLAVE_VOL
   start force
   ```

7. Verify the status of the created session, using the following command:

   ```
   # gluster volume geo-replication MASTER_VOL SLAVE_HOST::SLAVE_VOL
   status
   ```

10.5.2. Starting Geo-replication for a New Brick on an Existing Node

When adding a brick to the volume on an existing node in the trusted storage pool with a geo-replication session running, the geo-replication daemon on that particular node will automatically be restarted. The new brick will then be recognized by the geo-replication daemon. This is an automated process and no configuration changes are required.

10.5.3. Starting Geo-replication for a New Volume

To create and start a geo-replication session between a new volume added to the master cluster and a new volume added to the slave cluster, you must perform the following steps:

Prerequisites

- There must be key-based SSH authentication without a password access between the master volume node and the slave volume node.

1. Create the geo-replication session using the following command:

   ```
   # gluster volume geo-replication MASTER_VOL SLAVE_HOST::SLAVE_VOL
   create
   ```

   For example:

   ```
   # gluster volume geo-replication Volume1 storage.backup.com::slave-vol
   create
   ```

   Note

   This command performs the slave verification, which includes checking for a valid slave URL, valid slave volume, and available space on the slave.

2. Configure the meta-volume for geo-replication:

   ```
   # gluster volume geo-replication MASTER_VOL SLAVE_HOST::SLAVE_VOL
   config use_meta_volume true
   ```

   For example:

   ```
   # gluster volume geo-replication Volume1 storage.backup.com::slave-vol
   config use_meta_volume true
   ```

   For more information on configuring meta-volume, see Section 10.3.5, “Configuring a Meta-Volume”.

3. Start the geo-replication session between the slave and master, using the following command:

   ```
   # gluster volume geo-replication MASTER_VOL SLAVE_HOST::SLAVE_VOL
   start
   ```

4. Verify the status of the created session, using the following command:

   ```
   # gluster volume geo-replication MASTER_VOL SLAVE_HOST::SLAVE_VOL
   status
   ```

10.6. Scheduling Geo-replication as a Cron Job

Cron is a daemon that can be used to schedule the execution of recurring tasks according to a combination of the time, day of the month, month, day of the week, and week. Cron assumes that the system is ON continuously. If the system is not ON when a task is scheduled, it is not executed. A script is provided to run geo-replication only when required or to schedule geo-replication to run during low I/O.

For more information on installing Cron and configuring Cron jobs, see Automating System Tasks in the *Red Hat Enterprise Linux 7 System Administrator's Guide*.

The script provided to schedule the geo-replication session, performs the following:

1. Stops the geo-replication session, if started
2. Starts the geo-replication session
3. Sets the Checkpoint
4. Checks the status of checkpoint until it is complete
5. After the checkpoint is complete, stops the geo-replication session

Run geo-replication Session

To run a geo-reolication session only when required, run the following script:

```
# python /usr/share/glusterfs/scripts/schedule_georep.py MASTERVOL SLAVEHOST
SLAVEVOL
```

For example,

```
# python /usr/share/glusterfs/scripts/schedule_georep.py Volume1
storage.backup.com slave-vol
```

Run the following command to view the help:

```
# python /usr/share/glusterfs/scripts/schedule_georep.py --help
```

Schedule a Cron Job

To schedule geo-replication to run automatically using Cron:

```
minute hour day month day-of-week directory_and_script-to-execute MASTERVOL
SLAVEHOST SLAVEVOL >> log_file_for_script_output
```

For example, to run geo-replication daily at 20:30 hours, run the following:

```
30 20 * * * root python /usr/share/glusterfs/scripts/schedule_georep.py --
no-color Volume1 storage.backup.com slave-vol >>
/var/log/glusterfs/schedule_georep.log 2>&1
```

10.7. Disaster Recovery

Red Hat Gluster Storage provides geo-replication failover and failback capabilities for disaster recovery. If the master goes offline, you can perform a **`failover`** procedure so that a slave can replace the master. When this happens, all the I/O operations, including reads and writes, are done on the slave which is now acting as the master. When the original master is back online, you can perform a **`failback`** procedure on the original slave so that it synchronizes the differences back to the original master.

10.7.1. Failover: Promoting a Slave to Master

If the master volume goes offline, you can promote a slave volume to be the master, and start using that volume for data access.

1. Disable read-only on the slave volume by running the following command:

   ```
   # gluster volume set VOLNAME features.read-only off
   ```

2. Run the following commands on the slave machine to promote it to be the master:

   ```
   # gluster volume set VOLNAME geo-replication.indexing on
    # gluster volume set VOLNAME changelog on
   ```

   For example

   ```
   # gluster volume set slave-vol geo-replication.indexing on
    volume set: success
    # gluster volume set slave-vol changelog on
    volume set: success
   ```

You can now configure applications to use the slave volume for I/O operations.

10.7.2. Failback: Resuming Master and Slave back to their Original State

When the original master is back online, you can perform the following procedure on the original slave so that it synchronizes the differences back to the original master:

1. Stop the existing geo-rep session from original master to orginal slave using the following command:

   ```
   # gluster volume geo-replication  ORIGINAL_MASTER_VOL
   ORIGINAL_SLAVE_HOST::ORIGINAL_SLAVE_VOL stop force
   ```

   For example,

   ```
   # gluster volume geo-replication  Volume1 storage.backup.com::slave-
   vol stop force
   Stopping geo-replication session between Volume1 and
   storage.backup.com::slave-vol has been successful
   ```

2. Create a new geo-replication session with the original slave as the new master, and the original master as the new slave with **`force`** option. Detailed information on creating geo-replication session is available at: .

   a. Section 10.3.3, “Prerequisites”

   b. Section 10.3.4.1, “Setting Up your Environment for Geo-replication Session”

   c. Section 10.3.5, “Configuring a Meta-Volume”

3. Start the special synchronization mode to speed up the recovery of data from slave. This option adds

capability to geo-replication to ignore the files created before enabling **indexing** option. With this option, geo-replication will synchronize only those files which are created after making Slave volume as Master volume.

```
# gluster volume geo-replication ORIGINAL_SLAVE_VOL
ORIGINAL_MASTER_HOST::ORIGINAL_MASTER_VOL config special-sync-mode
recover
```

For example,

```
# gluster volume geo-replication  slave-vol master.com::Volume1 config
special-sync-mode recover
geo-replication config updated successfully
```

4. Disable the gfid-conflict-resolution option:

```
# gluster volume geo-replication ORIGINAL_SLAVE_VOL
ORIGINAL_MASTER_HOST::ORIGINAL_MASTER_VOL config gfid-conflict-
resolution false
```

For example,

```
# gluster volume geo-replication slave-vol master.com::Volume1 config
gfid-conflict-resolution false
geo-replication config updated successfully
```

5. Start the new geo-replication session using the following command:

```
# gluster volume geo-replication ORIGINAL_SLAVE_VOL
ORIGINAL_MASTER_HOST::ORIGINAL_MASTER_VOL start
```

For example,

```
# gluster volume geo-replication slave-vol master.com::Volume1 start
Starting geo-replication session between slave-vol and
master.com::Volume1 has been successful
```

6. Stop the I/O operations on the original slave and set the checkpoint. By setting a checkpoint, synchronization information is available on whether the data that was on the master at that point in time has been replicated to the slaves.

```
# gluster volume geo-replication ORIGINAL_SLAVE_VOL
ORIGINAL_MASTER_HOST::ORIGINAL_MASTER_VOL config checkpoint now
```

For example,

```
# gluster volume geo-replication slave-vol master.com::Volume1 config
checkpoint now
geo-replication config updated successfully
```

7. Checkpoint completion ensures that the data from the original slave is restored back to the original master. But since the IOs were stopped at slave before checkpoint was set, we need to touch the slave mount for checkpoint to be completed

```
# touch orginial_slave_mount
```

```
# gluster volume geo-replication ORIGINAL_SLAVE_VOL
ORIGINAL_MASTER_HOST::ORIGINAL_MASTER_VOL status detail
```

For example,

```
# touch /mnt/gluster/slavevol
# gluster volume geo-replication slave-vol master.com::Volume1 status
detail
```

8. After the checkpoint is complete, stop and delete the current geo-replication session between the original slave and original master

```
# gluster volume geo-replication ORIGINAL_SLAVE_VOL
ORIGINAL_MASTER_HOST::ORIGINAL_MASTER_VOL stop
```

```
# gluster volume geo-replication ORIGINAL_SLAVE_VOL
ORIGINAL_MASTER_HOST::ORIGINAL_MASTER_VOL delete
```

For example,

```
# gluster volume geo-replication slave-vol master.com::Volume1 stop
Stopping geo-replication session between slave-vol and
master.com::Volume1 has been successful

# gluster volume geo-replication slave-vol master.com::Volume1 delete
geo-replication command executed successfully
```

9. Disable read-only on the master volume by running the following command:

```
# gluster volume set VOLNAME features.read-only off
```

10. Reset the options that were set for promoting the slave volume as the master volume by running the following commands:

```
# gluster volume reset ORIGINAL_SLAVE_VOL geo-replication.indexing
force
# gluster volume reset ORIGINAL_SLAVE_VOL changelog
```

For example,

```
# gluster volume reset slave-vol geo-replication.indexing force
volume set: success

# gluster volume reset slave-vol changelog
volume set: success
```

11. Resume the original roles by starting the geo-rep session from the original master using the following command:

```
# gluster volume geo-replication ORIGINAL_MASTER_VOL
```

```
ORIGINAL_SLAVE_HOST::ORIGINAL_SLAVE_VOL start
```

```
# gluster volume geo-replication Volume1 storage.backup.com::slave-vol
start
Starting geo-replication session between slave-vol  and
master.com::Volume1 been successful
```

10.8. Creating a Snapshot of Geo-replicated Volume

The Red Hat Gluster Storage Snapshot feature enables you to create point-in-time copies of Red Hat Gluster Storage volumes, which you can use to protect data. You can create snapshots of Geo-replicated volumes.

For information on prerequisites, creating, and restoring snapshots of geo-replicated volume, see Chapter 8, *Managing Snapshots*. Creation of a snapshot when geo-replication session is live is not supported and creation of snapshot in this scenario will display the following error:

```
# gluster snapshot create snap1 master
snapshot create: failed: geo-replication session is running for the volume
master. Session needs to be stopped before taking a snapshot.
Snapshot command failed
```

.

You must ensure to pause the geo-replication session before creating snapshot and resume geo-replication session after creating the snapshot. Information on restoring geo-replicated volume is also available in the *Managing Snapshots* chapter.

10.9. Example - Setting up Cascading Geo-replication

This section provides step by step instructions to set up a cascading geo-replication session. The configuration of this example has three volumes and the volume names are master-vol, interimmaster-vol, and slave-vol.

1. Verify that your environment matches the minimum system requirements listed in Section 10.3.3, “Prerequisites”.

2. Determine the appropriate deployment scenario. For more information on deployment scenarios, see Section 10.3.1, “Exploring Geo-replication Deployment Scenarios”.

3. Configure the environment and create a geo-replication session between master-vol and interimmaster-vol.

   a. Create a common pem pub file, run the following command on the master node where the key-based SSH authentication connection is configured:

   ```
   # gluster system:: execute gsec_create
   ```

   b. Create the geo-replication session using the following command. The push-pem option is needed to perform the necessary pem-file setup on the interimmaster nodes.

   ```
   # gluster volume geo-replication master-vol
   interimhost.com::interimmaster-vol create
   push-pem
   ```

c. Verify the status of the created session by running the following command:

```
# gluster volume geo-replication master-vol
interimhost::interimmaster-vol status
```

4. Configure the meta-volume for geo-replication:

   ```
   # gluster volume geo-replication master-vol
   interimhost.com::interimmaster-vol config use_meta_volume true
   ```

   For more information on configuring meta-volume, see Section 10.3.5, “Configuring a Meta-Volume”.

5. Start a Geo-replication session between the hosts:

   ```
   # gluster volume geo-replication master-vol
   interimhost.com::interimmaster-vol start
   ```

   This command will start distributed geo-replication on all the nodes that are part of the master volume. If a node that is part of the master volume is down, the command will still be successful. In a replica pair, the geo-replication session will be active on any of the replica nodes, but remain passive on the others. After executing the command, it may take a few minutes for the session to initialize and become stable.

6. Verifying the status of geo-replication session by running the following command:

   ```
   # gluster volume geo-replication master-vol
   interimhost.com::interimmaster-vol status
   ```

7. Create a geo-replication session between interimmaster-vol and slave-vol.

   a. Create a common pem pub file by running the following command on the interimmaster master node where the key-based SSH authentication connection is configured:

   ```
   # gluster system:: execute gsec_create
   ```

   b. On interimmaster node, create the geo-replication session using the following command. The push-pem option is needed to perform the necessary pem-file setup on the slave nodes.

   ```
   # gluster volume geo-replication interimmaster-vol
   slave_host.com::slave-vol create push-pem
   ```

   c. Verify the status of the created session by running the following command:

   ```
   # gluster volume geo-replication interrimmaster-vol
   slave_host::slave-vol status
   ```

8. Configure the meta-volume for geo-replication:

   ```
   # gluster volume geo-replication interrimmaster-vol slave_host::slave-
   vol config use_meta_volume true
   ```

   For more information on configuring meta-volume, see Section 10.3.5, “Configuring a Meta-Volume”.

9. Start a geo-replication session between interrimaster-vol and slave-vol by running the following command:

   ```
   # gluster volume geo-replication interrimmaster-vol
   slave_host.com::slave-vol start
   ```

10. Verify the status of geo-replication session by running the following:

    ```
    # gluster volume geo-replication interrimmaster-vol
    slave_host.com::slave-vol status
    ```

10.10. Recommended Practices

Manually Setting the Time

If you have to change the time on the bricks manually, then the geo-replication session and indexing must be disabled when setting the time on all the bricks. All bricks in a geo-replication environment must be set to the same time, as this avoids the out-of-time sync issue described in Section 10.3.4.1, "Setting Up your Environment for Geo-replication Session". Bricks not operating on the same time setting, or changing the time while the geo-replication is running, will corrupt the geo-replication index. The recommended way to set the time manually is using the following procedure.

Manually Setting the Time on Bricks in a Geo-replication Environment

1. Stop geo-replication between the master and slave, using the following command:

   ```
   # gluster volume geo-replication MASTER_VOL SLAVE_HOST::SLAVE_VOL stop
   ```

2. Stop geo-replication indexing, using the following command:

   ```
   # gluster volume set MASTER_VOL geo-replication.indexing off
   ```

3. Set a uniform time on all the bricks.

4. Restart the geo-replication sessions, using the following command:

   ```
   # gluster volume geo-replication MASTER_VOL SLAVE_HOST::SLAVE_VOL
   start
   ```

Performance Tuning

When the following option is set, it has been observed that there is an increase in geo-replication performance. On the slave volume, run the following command:

```
# gluster volume set SLAVE_VOL batch-fsync-delay-usec 0
```

Initially Replicating Large Volumes to a Remote Slave Locally using a LAN

For replicating large volumes to a slave in a remote location, it may be useful to do the initial replication to disks locally on a local area network (LAN), and then physically transport the disks to the remote location. This eliminates the need of doing the initial replication of the whole volume over a slower and more expensive wide area network (WAN) connection. The following procedure provides instructions for setting up a local geo-replication session, physically transporting the disks to the remote location, and then setting up geo-replication over a WAN.

Initially Replicating to a Remote Slave Locally using a LAN

1. Create a geo-replication session locally within the LAN. For information on creating a geo-replication session, see Section 10.3.4.1, “Setting Up your Environment for Geo-replication Session”.

   > **Important**
   >
   > You must remember the order in which the bricks/disks are specified when creating the slave volume. This information is required later for configuring the remote geo-replication session over the WAN.

2. Ensure that the initial data on the master is synced to the slave volume. You can verify the status of the synchronization by using the **status** command, as shown in Section 10.4.3, “Displaying Geo-replication Status Information”.

3. Stop and delete the geo-replication session.

   For information on stopping and deleting the the geo-replication session, see Section 10.4.5, “Stopping a Geo-replication Session” and Section 10.4.6, “Deleting a Geo-replication Session”.

   > **Important**
   >
   > You must ensure that there are no stale files in **/var/lib/glusterd/geo-replication/**.

4. Stop and delete the slave volume.

   For information on stopping and deleting the volume, see Section 11.13, “Stopping Volumes” and Section 11.14, “Deleting Volumes”.

5. Remove the disks from the slave nodes, and physically transport them to the remote location. Make sure to remember the order in which the disks were specified in the volume.

6. At the remote location, attach the disks and mount them on the slave nodes. Make sure that the file system or logical volume manager is recognized, and that the data is accessible after mounting it.

7. Configure a trusted storage pool for the slave using the **peer probe** command.

   For information on configuring a trusted storage pool, see Chapter 4, *Adding Servers to the Trusted Storage Pool*.

8. Delete the glusterFS-related attributes on the bricks. This should be done before creating the volume. You can remove the glusterFS-related attributes by running the following command:

   ```
   # for i in `getfattr -d -m . ABSOLUTE_PATH_TO_BRICK 2>/dev/null | grep
   trusted | awk -F = '{print $1}'`; do setfattr -x $i
   ABSOLUTE_PATH_TO_BRICK; done
   ```

   Run the following command to ensure that there are no **xattrs** still set on the brick:

```
# getfattr -d -m . ABSOLUTE_PATH_TO_BRICK
```

9. After creating the trusted storage pool, create the Red Hat Gluster Storage volume with the same configuration that it had when it was on the LAN. For information on creating volumes, see Chapter 5, *Setting Up Storage Volumes*.

   Important

   Make sure to specify the bricks in same order as they were previously when on the LAN. A mismatch in the specification of the brick order may lead to data loss or corruption.

10. Start and mount the volume, and check if the data is intact and accessible.

    For information on starting and mounting volumes, see Section 5.11, “Starting Volumes” and Chapter 6, *Creating Access to Volumes*.

11. Configure the environment and create a geo-replication session from the master to this remote slave.

    For information on configuring the environment and creating a geo-replication session, see Section 10.3.4.1, “Setting Up your Environment for Geo-replication Session”.

12. Start the geo-replication session between the master and the remote slave.

    For information on starting the geo-replication session, see Section 10.4, “Starting Geo-replication”.

13. Use the **status** command to verify the status of the session, and check if all the nodes in the session are stable.

    For information on the **status**, see Section 10.4.3, “Displaying Geo-replication Status Information”.

10.11. Troubleshooting Geo-replication

This section describes the most common troubleshooting scenarios related to geo-replication.

10.11.1. Tuning Geo-replication performance with Change Log

There are options for the change log that can be configured to give better performance in a geo-replication environment.

The **rollover-time** option sets the rate at which the change log is consumed. The default rollover time is 60 seconds, but it can be configured to a faster rate. A recommended rollover-time for geo-replication is 10-15 seconds. To change the **rollover-time** option, use following the command:

```
# gluster volume set VOLNAME rollover-time 15
```

The **fsync-interval** option determines the frequency that updates to the change log are written to disk. The default interval is 0, which means that updates to the change log are written synchronously as they occur, and this may negatively impact performance in a geo-replication environment. Configuring **fsync-interval** to a non-zero value will write updates to disk asynchronously at the specified interval. To change the **fsync-interval** option, use following the command:

```
# gluster volume set VOLNAME fsync-interval 3
```

10.11.2. Triggering Explicit Sync on Entries

Geo-replication provides an option to explicitly trigger the sync operation of files and directories. A virtual extended attribute **`glusterfs.geo-rep.trigger-sync`** is provided to accomplish the same.

```
# setfattr -n glusterfs.geo-rep.trigger-sync -v "1" <file-path>
```

The support of explicit trigger of sync is supported only for directories and regular files.

10.11.3. Synchronization Is Not Complete

Situation

The geo-replication status is displayed as **`Stable`**, but the data has not been completely synchronized.

Solution

A full synchronization of the data can be performed by erasing the index and restarting geo-replication. After restarting geo-replication, it will begin a synchronization of the data using checksums. This may be a long and resource intensive process on large data sets. If the issue persists, contact Red Hat Support.

For more information about erasing the index, see Section 11.1, “Configuring Volume Options”.

10.11.4. Issues with File Synchronization

Situation

The geo-replication status is displayed as **`Stable`**, but only directories and symlinks are synchronized. Error messages similar to the following are in the logs:

```
[2011-05-02 13:42:13.467644] E [master:288:regjob] GMaster: failed to sync
./some_file`
```

Solution

Geo-replication requires **`rsync`** v3.0.0 or higher on the host and the remote machines. Verify if you have installed the required version of **`rsync`**.

10.11.5. Geo-replication Status is Often `Faulty`

Situation

The geo-replication status is often displayed as **`Faulty`**, with a backtrace similar to the following:

```
012-09-28 14:06:18.378859] E [syncdutils:131:log_raise_exception] <top>:
FAIL: Traceback (most recent call last): File
"/usr/local/libexec/glusterfs/python/syncdaemon/syncdutils.py", line 152, in
twraptf(*aa) File "/usr/local/libexec/glusterfs/python/syncdaemon/repce.py",
line 118, in listen rid, exc, res = recv(self.inf) File
"/usr/local/libexec/glusterfs/python/syncdaemon/repce.py", line 42, in recv
return pickle.load(inf) EOFError
```

Solution

This usually indicates that RPC communication between the master gsyncd module and slave gsyncd module is broken. Make sure that the following prerequisites are met:

- Key-based SSH authentication is set up properly between the host and remote machines.
- FUSE is installed on the machines. The geo-replication module mounts Red Hat Gluster Storage volumes using FUSE to sync data.

10.11.6. Intermediate Master is in a Faulty State

Situation

In a cascading environment, the intermediate master is in a faulty state, and messages similar to the following are in the log:

```
raise RuntimeError ("aborting on uuid change from %s to %s" % \
RuntimeError: aborting on uuid change from af07e07c-427f-4586-ab9f-
4bf7d299be81 to de6b5040-8f4e-4575-8831-c4f55bd41154
```

Solution

In a cascading configuration, an intermediate master is loyal to its original primary master. The above log message indicates that the geo-replication module has detected that the primary master has changed. If this change was deliberate, delete the **`volume-id`** configuration option in the session that was initiated from the intermediate master.

10.11.7. Remote gsyncd Not Found

Situation

The master is in a faulty state, and messages similar to the following are in the log:

```
[2012-04-04 03:41:40.324496] E [resource:169:errfail] Popen: ssh> bash:
/usr/local/libexec/glusterfs/gsyncd: No such file or directory
```

Solution

The steps to configure a SSH connection for geo-replication have been updated. Use the steps as described in Section 10.3.4.1, “Setting Up your Environment for Geo-replication Session”

Chapter 11. Managing Red Hat Gluster Storage Volumes

This chapter describes how to perform common volume management operations on the Red Hat Gluster Storage volumes.

11.1. Configuring Volume Options

Note

Volume options can be configured while the trusted storage pool is online.

The current settings for a volume can be viewed using the following command:

```
# gluster volume info VOLNAME
```

Volume options can be configured using the following command:

```
# gluster volume set VOLNAME OPTION PARAMETER
```

For example, to specify the performance cache size for **test-volume**:

```
# gluster volume set test-volume performance.cache-size 256MB
Set volume successful
```

11.2. Setting Multiple Volume Option

A group configuration file is a file used to define and customize volume options. There are some predefined group configuration files for specific workload patterns like negative lookup cache, virtualization, metadata cache and gluster-block.

The parameters defined in the file can then be applied to a volume as a group, rather than setting one parameter at a time.

Creating a group configuration file

1. Create a new file in the **/var/lib/glusterd/groups/** directory.

   ```
   # touch /var/lib/glusterd/groups/filename
   ```

2. Add the parameters and values that you want to set on the volume to the created file as key-value pairs, placing each parameter on a new line:

   ```
   domain1.key1=value1
   domain1.key2=value2
   domain2.key3=value3
   ```

   For example,

```
changelog.changelog=on
client.event-threads=6
cluster.brick-multiplex=on
```

Adding configurations to volumes

Run the following command to apply the configurations in the group file to specific volumes:

```
# gluster volume set volname group filename
```

For example,

```
# gluster volume set volume1 group virt
# gluster volume set volume2 group virt
# gluster volume set volume3 group dbgroup
```

> **Note**
>
> The configuration file created should be placed in all the hosts of the trusted storage pool under */var/lib/glusterd/groups/*. This can be achieved with the help of gdeploy configuration file.

11.3. Supported Volume Options

The following table lists available volume options along with their description and default value.

> **Important**
>
> The default values are subject to change, and may not be the same for all versions of Red Hat Gluster Storage.

Table 11.1. Volume Options

Option	Value Description	Allowed Values	Default Value
auth.allow	IP addresses or hostnames of the clients which are allowed to access the volume.	Valid hostnames or IP addresses, which includes wild card patterns including *****. For example, **192.168.1.***. A list of comma separated addresses is acceptable, but a single hostname must not exceed 256 characters.	* (allow all)

Option	Value Description	Allowed Values	Default Value
auth.reject	IP addresses or hostnames of FUSE clients that are denied access to a volume. For NFS access control, use **`nfs.rpc-auth-*`** options instead.	Valid hostnames or IP addresses, which includes wild card patterns including **`*`**. For example, **`192.168.1.*`**. A list of comma separated addresses is acceptable, but a single hostname must not exceed 256 characters.	none (reject none)
changelog	Enables the changelog translator to record all the file operations.	on | off	off
client.event-threads	Specifies the number of network connections to be handled simultaneously by the client processes accessing a Red Hat Gluster Storage node.	1 - 32	2
cluster.background-self-heal-count	The maximum number of heal operations that can occur simultaneously. Requests in excess of this number are stored in a queue whose length is defined by **`cluster.heal-wait-queue-leng`**.	0–256	8

Option	Value Description	Allowed Values	Default Value
cluster.brick-multiplex	Available as of Red Hat Gluster Storage 3.3 and later. Controls whether to use brick multiplexing on all volumes. Red Hat recommends restarting volumes after enabling or disabling brick multiplexing. When set to **`off`** (the default), each brick has its own process and uses its own port. When set to **on**, bricks that are compatible with each other use the same process and the same port. This reduces per-brick memory usage and port consumption. Brick compatibility is determined at volume start, and depends on volume options shared between bricks. When multiplexing is enabled, restart volumes whenever volume configuration is changed in order to maintain the compatibility of the bricks grouped under a single process.	on | off	off

Option	Value Description	Allowed Values	Default Value
cluster.consistent-metadata	If set to **on**, the readdirp function in Automatic File Replication feature will always fetch metadata from their respective read children as long as it holds the good copy (the copy that does not need healing) of the file/directory. However, this could cause a reduction in performance where readdirps are involved. This option requires that the volume is remounted on the client to take effect.	on | off	off
cluster.granular-entry-heal	If set to enable, stores more granular information about the entries which were created or deleted from a directory while a brick in a replica was down. This helps in faster self-heal of directories, especially in use cases where directories with large number of entries are modified by creating or deleting entries. If set to disable, it only stores that the directory needs heal without information about what entries within the directories need to be healed, and thereby requires entire directory crawl to identify the changes.	enable | disable	disable

Important

Execute the **`gluster volume set VOLNAME cluster.granular-entry-heal [enable | disable]`** command only if the volume is in **`Created`** state. If the volume is in any other state other than **`Created`**, for example, **`Started`**, **`Stopped`**, and so on, execute **`gluster volume heal VOLNAME granular-entry-heal [enable | disable]`** command to enable or disable **`granular-entry-heal`** option.

Option	Value Description	Allowed Values	Default Value
cluster.heal-wait-queue-leng	The maximum number of requests for heal operations that can be queued when heal operations equal to **`cluster.background-self-heal-count`** are already in progress. If more heal requests are made when this queue is full, those heal requests are ignored.	0-10000	128
cluster.lookup-optimize	If this option is set to **on**, when a hashed sub-volume does not return a lookup result, negative lookups are optimized by not continuing to look on non-hashed subvolumes. For existing volumes, any directories created after the upgrade will have lookup-optimize behavior enabled. Rebalance operation has to be performed on all existing directories before they can use the lookup optimization. For new volumes, the lookup-optimize behavior is enabled by default, except for the root of the volume. Run a rebalance operation in order to enable lookup-optimize for the root of the volume.	on|off	on (in version 3.4 onward)

Option	Value Description	Allowed Values	Default Value
cluster.max-bricks-per-process	The maximum number of bricks that can run on a single instance of glusterfsd process. As of Red Hat Gluster Storage 3.4 Batch 2 Update, the default value of this option is set to **`250`**. This provides better control of resource usage for container-based workloads. In earlier versions, the default value was **`0`**, which used a single process for all bricks on the node. Updating the value of this option does not affect currently running bricks. Restart the volume to change this setting for existing bricks.	0 to system maximum (any positive integer greater than 1)	250
cluster.min-free-disk	Specifies the percentage of disk space that must be kept free. This may be useful for non-uniform bricks.	Percentage of required minimum free disk space.	10%
cluster.op-version	Allows you to set the operating version of the cluster. The op-version number cannot be downgraded and is set for all volumes in the cluster. The op-version is not listed as part of gluster volume info command output.	30708 | 30712 | 31001 | 31101 | 31302 | 31303 | 31304 | 31305	Default value depends on Red Hat Gluster Storage version first installed. For Red Hat Gluster Storage 3.4 Batch 3 Update the value is set to 31305 for a new deployment.
cluster.read-freq-threshold	Specifies the number of reads, in a promotion/demotion cycle, that would mark a file HOT for promotion. Any file that has read hits less than this value will be considered as COLD and will be demoted.	0-20	0

Option	Value Description	Allowed Values	Default Value
cluster.self-heal-daemon	Specifies whether proactive self-healing on replicated volumes is activated.	on | off	on
cluster.data-self-heal	Specifies whether proactive data self-healing on replicated volumes is activated.	on | off	on
cluster.entry-self-heal	Specifies whether proactive self-healing of the contents of a directory on replicated volumes is activated.	on | off	on
cluster.metadata-self-heal	Specifies whether proactive metadata self-healing on replicated volumes is activated.	on | off	on

Important

By default, **cluster.data-self-heal**, **cluster.entry-self-heal**, and **cluster.metadata-self-heal** options are turned on. These options are for client-side, that is, mounts, healing of files. These will not affect the self-heal daemon. As a file is accessed healing gets triggered on it without waiting for the Self-heal-daemon to select and heal the file. This maximizes the client-side I/O throughput. It is recommended to turn these options off, with the help of Red Hat support, in the case of severe CPU or memory consumption. When turned off, self-heal is not triggered when the client accesses the file and the entire network bandwidth is available for I/O from the clients (client-side healing and accessing files pending heal consumed some). The self-heal daemon will still heal the file during its crawl.

cluster.server-quorum-ratio	Sets the quorum percentage for the trusted storage pool.	0 - 100	>50%
cluster.server-quorum-type	If set to server, this option enables the specified volume to participate in the server-side quorum. For more information on configuring the server-side quorum, see Section 11.15.1.1, “Configuring Server-Side Quorum”	none | server	none
cluster.shd-max-threads	Specifies the number of entries that can be self healed in parallel on each replica by self-heal daemon.	1 - 64	1

Option	Value Description	Allowed Values	Default Value
cluster.shd-max-threads	Specifies the number of entries that can be self healed in parallel on each replica by self-heal daemon.	1 - 64	1
cluster.shd-wait-qlength	Specifies the number of entries that must be kept in the queue for self-heal daemon threads to take up as soon as any of the threads are free to heal. This value should be changed based on how much memory self-heal daemon process can use for keeping the next set of entries that need to be healed.	1 - 655536	1024
cluster.shd-wait-qlength	Specifies the number of entries that must be kept in the dispersed subvolume's queue for self-heal daemon threads to take up as soon as any of the threads are free to heal. This value should be changed based on how much memory self-heal daemon process can use for keeping the next set of entries that need to be healed.	1 - 655536	1024
cluster.tier-demote-frequency	Specifies how frequently the tier daemon must check for files to demote.	1 - 172800 seconds	3600 seconds
cluster.tier-max-files	Specifies the maximum number of files that may be migrated in any direction from each node in a given cycle.	1-100000 files	10000
cluster.tier-max-mb	Specifies the maximum number of MB that may be migrated in any direction from each node in a given cycle.	1 -100000 (100 GB)	4000 MB

Option	Value Description	Allowed Values	Default Value
cluster.tier-mode	If set to **`cache`** mode, promotes or demotes files based on whether the cache is full or not, as specified with watermarks. If set to **`test`** mode, periodically demotes or promotes files automatically based on access.	test | cache	cache
cluster.tier-promote-frequency	Specifies how frequently the tier daemon must check for files to promote.	1- 172800 seconds	120 seconds
cluster.use-compound-fops	When enabled, write transactions that occur as part of Automatic File Replication are modified so that network round trips are reduced, improving performance.	on | off	off
cluster.watermark-hi	Upper percentage watermark for promotion. If hot tier fills above this percentage, no promotion will happen and demotion will happen with high probability.	1- 99 %	90%
cluster.watermark-low	Lower percentage watermark. If hot tier is less full than this, promotion will happen and demotion will not happen. If greater than this, promotion/demotion will happen at a probability relative to how full the hot tier is.	1- 99 %	75%
cluster.write-freq-threshold	Specifies the number of writes, in a promotion/demotion cycle, that would mark a file HOT for promotion. Any file that has write hits less than this value will be considered as COLD and will be demoted.	0-20	0
config.transport	Specifies the type of transport(s) volume would support communicating over.	tcp OR rdma OR tcp,rdma	tcp

Option	Value Description	Allowed Values	Default Value
diagnostics.brick-log-buf-size	The maximum number of unique log messages that can be suppressed until the timeout or buffer overflow, whichever occurs first on the bricks.	0 and 20 (0 and 20 included)	5
diagnostics.brick-log-flush-timeout	The length of time for which the log messages are buffered, before being flushed to the logging infrastructure (gluster or syslog files) on the bricks.	30 - 300 seconds (30 and 300 included)	120 seconds
diagnostics.brick-log-format	Allows you to configure the log format to log either with a message id or without one on the brick.	no-msg-id | with-msg-id	with-msg-id
diagnostics.brick-log-level	Changes the log-level of the bricks.	INFO | DEBUG | WARNING | ERROR | CRITICAL | NONE | TRACE	info
diagnostics.brick-sys-log-level	Depending on the value defined for this option, log messages at and above the defined level are generated in the syslog and the brick log files.	INFO | WARNING | ERROR | CRITICAL	CRITICAL
diagnostics.client-log-buf-size	The maximum number of unique log messages that can be suppressed until the timeout or buffer overflow, whichever occurs first on the clients.	0 and 20 (0 and 20 included)	5
diagnostics.client-log-flush-timeout	The length of time for which the log messages are buffered, before being flushed to the logging infrastructure (gluster or syslog files) on the clients.	30 - 300 seconds (30 and 300 included)	120 seconds
diagnostics.client-log-format	Allows you to configure the log format to log either with a message ID or without one on the client.	no-msg-id | with-msg-id	with-msg-id
diagnostics.client-log-level	Changes the log-level of the clients.	INFO | DEBUG | WARNING | ERROR | CRITICAL | NONE | TRACE	info

Option	Value Description	Allowed Values	Default Value
diagnostics.client-sys-log-level	Depending on the value defined for this option, log messages at and above the defined level are generated in the syslog and the client log files.	INFO | WARNING | ERROR | CRITICAL	CRITICAL
disperse.eager-lock	Before a file operation starts, a lock is placed on the file. The lock remains in place until the file operation is complete. After the file operation completes, if eager-lock is on, the lock remains in place either until lock contention is detected, or for 1 second in order to check if there is another request for that file from the same client. If eager-lock is off, locks release immediately after file operations complete, improving performance for some operations, but reducing access efficiency.	on | off	on
disperse.other-eager-lock	This option is equivalent to the **`disperse.eager-lock`** option but applicable only for non regular files. When multiple clients access a particular directory, disabling disperse.other-eager-lockoption for the volume can improve performance for directory access without compromising performance of I/O's for regular files.	on | off	off
disperse.shd-max-threads	Specifies the number of entries that can be self healed in parallel on each disperse subvolume by self-heal daemon.	1 - 64	1

Option	Value Description	Allowed Values	Default Value
disperse.shd-wait-qlength	Specifies the number of entries that must be kept in the dispersed subvolume's queue for self-heal daemon threads to take up as soon as any of the threads are free to heal. This value should be changed based on how much memory self-heal daemon process can use for keeping the next set of entries that need to be healed.	1 - 655536	1024
features.ctr_link_consistency	Enables a crash consistent way of recording hardlink updates by Change Time Recorder translator. When recording in a crash consistent way the data operations will experience more latency.	on | off	off
features.ctr-enabled	Enables Change Time Recorder (CTR) translator for a tiered volume. This option is used in conjunction with features.record-counters option to enable recording write and read heat counters.	on | off	on
features.quota-deem-statfs	When this option is set to on, it takes the quota limits into consideration while estimating the filesystem size. The limit will be treated as the total size instead of the actual size of filesystem.	on | off	on
features.read-only	Specifies whether to mount the entire volume as read-only for all the clients accessing it.	on | off	off

Option	Value Description	Allowed Values	Default Value
features.record-counters	If set to enabled, cluster.write-freq-thresholdand cluster.read-freq-thresholdoptions defines the number of writes and reads to a given file that are needed before triggering migration.	on | off	on
features.shard	Enables or disables sharding on the volume. Affects files created after volume configuration.	enable | disable	disable
features.shard-block-size	Specifies the maximum size of file pieces when sharding is enabled. Affects files created after volume configuration.	512MB	512MB
geo-replication.indexing	Enables the marker translator to track the changes in the volume.	on | off	off
network.ping-timeout	The time the client waits for a response from the server. If a timeout occurs, all resources held by the server on behalf of the client are cleaned up. When the connection is reestablished, all resources need to be reacquired before the client can resume operations on the server. Additionally, locks are acquired and the lock tables are updated. A reconnect is a very expensive operation and must be avoided.	42 seconds	42 seconds
nfs.acl	Disabling nfs.acl will remove support for the NFSACL sideband protocol. This is enabled by default.	enable | disable	enable

Option	Value Description	Allowed Values	Default Value
nfs.addr-namelookup	Specifies whether to lookup names for incoming client connections. In some configurations, the name server can take too long to reply to DNS queries, resulting in timeouts of mount requests. This option can be used to disable name lookups during address authentication. Note that disabling name lookups will prevent you from using hostnames in nfs.rpc-auth-*options.	on | off	off
nfs.disable	Specifies whether to disable NFS exports of individual volumes.	on | off	off
nfs.enable-ino32	For nfs clients or applciatons that do not support 64-bit inode numbers, use this option to make NFS return 32-bit inode numbers instead. Disabled by default, so NFS returns 64-bit inode numbers. This value is global and applies to all the volumes in the trusted storage pool.	enable | disable	disable
nfs.export-volumes	Enables or disables exporting entire volumes. If this option is disabled and the **`nfs.export-diroption`** is enabled, you can set subdirectories as the only exports.	on | off	on

Option	Value Description	Allowed Values	Default Value
nfs.mount-rmtab	Path to the cache file that contains a list of NFS-clients and the volumes they have mounted. Change the location of this file to a mounted (with glusterfs-fuse, on all storage servers) volume to gain a trusted pool wide view of all NFS-clients that use the volumes. The contents of this file provide the information that can get obtained with the **`showmount`** command.	Path to a directory	/var/lib/glusterd/nfs/rmtab
nfs.mount-udp	Enable UDP transport for the MOUNT sideband protocol. By default, UDP is not enabled, and MOUNT can only be used over TCP. Some NFS-clients (certain Solaris, HP-UX and others) do not support MOUNT over TCP and enabling nfs.mount-udpmakes it possible to use NFS exports provided by Red Hat Gluster Storage.	disable | enable	disable
nfs.nlm	By default, the Network Lock Manager (NLMv4) is enabled. Use this option to disable NLM. Red Hat does not recommend disabling this option.	on|off	on
nfs.port	Associates glusterFS NFS with a non-default port.	1025-65535	38465- 38467
nfs.ports-insecure	Allows client connections from unprivileged ports. By default only privileged ports are allowed. This is a global setting for allowing insecure ports for all exports using a single option.	on | off	off

Option	Value Description	Allowed Values	Default Value
nfs.rdirplus	The default value is on. When this option is turned off, NFS falls back to standard readdir instead of readdirp. Turning this off would result in more lookup and stat requests being sent from the client which may impact performance.	on|off	on
nfs.rpc-auth-allow IP_ADRESSES	A comma separated list of IP addresses allowed to connect to the server. By default, all clients are allowed.	Comma separated list of IP addresses	accept all
nfs.rpc-auth-reject IP_ADRESSES	A comma separated list of addresses not allowed to connect to the server. By default, all connections are allowed.	Comma separated list of IP addresses	reject none
nfs.server-aux-gids	When enabled, the NFS-server will resolve the groups of the user accessing the volume. NFSv3 is restricted by the RPC protocol (AUTH_UNIX/AUTH_SYS header) to 16 groups. By resolving the groups on the NFS-server, this limits can get by-passed.	on|off	off
nfs.transport-type	Specifies the transport used by GlusterFS NFS server to communicate with bricks.	tcp OR rdma	tcp
open-behind	It improves the application's ability to read data from a file by sending success notifications to the application whenever it receives an open call.	on | off	on
performance.cache-max-file-size	Sets the maximum file size cached by the io-cache translator. Can be specified using the normal size descriptors of KB, MB, GB, TB, or PB (for example, 6 GB).	Size in bytes, or specified using size descriptors.	2 ^ 64-1 bytes

Option	Value Description	Allowed Values	Default Value
performance.cache-min-file-size	Sets the minimum file size cached by the io-cache translator. Can be specified using the normal size descriptors of KB, MB, GB, TB, or PB (for example, 6 GB).	Size in bytes, or specified using size descriptors.	0
performance.cache-refresh-timeout	The number of seconds cached data for a file will be retained. After this timeout, data re-validation will be performed.	0 - 61 seconds	1 second
performance.cache-size	Size of the read cache.	Size in bytes, or specified using size descriptors.	32 MB
performance.client-io-threads	Improves performance for parallel I/O from a single mount point for dispersed (erasure-coded) volumes by allowing up to 16 threads to be used in parallel. When enabled, 1 thread is used by default, and further threads up to the maximum of 16 are created as required by client workload. This is useful for dispersed and distributed dispersed volumes. This feature is not recommended for distributed, replicated or distributed-replicated volumes. It is disabled by default on replicated and distributed-replicated volume types.	on | off	on, except for replicated and distributed-replicated volumes
performance.flush-behind	Specifies whether the write-behind translator performs flush operations in the background by returning (false) success to the application before flush file operations are sent to the backend file system.	on | off	on
performance.io-thread-count	The number of threads in the I/O threads translator.	0 - 65	16

Option	Value Description	Allowed Values	Default Value
performance.lazy-open	This option requires open-behind to be on. Perform an open in the backend only when a necessary FOP arrives (for example, write on the file descriptor, unlink of the file). When this option is disabled, perform backend open immediately after an unwinding open.	Yes/No	Yes
performance.md-cache-timeout	The time period in seconds which controls when metadata cache has to be refreshed. If the age of cache is greater than this time-period, it is refreshed. Every time cache is refreshed, its age is reset to **`0`**.	0-600 seconds	1 second
performance.nfs-strict-write-ordering	Specifies whether to prevent later writes from overtaking earlier writes for NFS, even if the writes do not relate to the same files or locations.	on | off	off
performance.nfs.flush-behind	Specifies whether the write-behind translator performs flush operations in the background for NFS by returning (false) success to the application before flush file operations are sent to the backend file system.	on | off	on

Option	Value Description	Allowed Values	Default Value
performance.nfs.strict-o-direct	Specifies whether to attempt to minimize the cache effects of I/O for a file on NFS. When this option is enabled and a file descriptor is opened using the O_DIRECT flag, write-back caching is disabled for writes that affect that file descriptor. When this option is disabled, O_DIRECT has no effect on caching. This option is ignored if **`performance.write-behind`** is disabled.	on | off	off
performance.nfs.write-behind-trickling-writes	Enables and disables trickling-write strategy for the write-behind translator for NFS clients.	on | off	on
performance.nfs.write-behind-window-size	Specifies the size of the write-behind buffer for a single file or inode for NFS.	512 KB - 1 GB	1 MB
performance.quick-read	To enable/disable quick-read translator in the volume.	on | off	on
performance.rda-cache-limit	The value specified for this option is the maximum size of cache consumed by the readdir-ahead translator. This value is global and the total memory consumption by readdir-ahead is capped by this value, irrespective of the number/size of directories cached.	0-1GB	10MB
performance.rda-request-size	The value specified for this option will be the size of buffer holding directory entries in readdirp response.	4KB-128KB	128KB
performance.resync-failed-syncs-after-fsync	If syncing cached writes that were issued before an fsync operation fails, this option configures whether to reattempt the failed sync operations.	on | off	off

Option	Value Description	Allowed Values	Default Value
performance.strict-o-direct	Specifies whether to attempt to minimize the cache effects of I/O for a file. When this option is enabled and a file descriptor is opened using the O_DIRECT flag, write-back caching is disabled for writes that affect that file descriptor. When this option is disabled, O_DIRECT has no effect on caching. This option is ignored if **`performance.write-behind`** is disabled.	on | off	off
performance.strict-write-ordering	Specifies whether to prevent later writes from overtaking earlier writes, even if the writes do not relate to the same files or locations.	on | off	off
performance.use-anonymous-fd	This option requires open-behind to be on. For read operations, use anonymous file descriptor when the original file descriptor is open-behind and not yet opened in the backend.	Yes | No	Yes
performance.write-behind	Enables and disables write-behind translator.	on | off	on
performance.write-behind-trickling-writes	Enables and disables trickling-write strategy for the write-behind translator for FUSE clients.	on | off	on
performance.write-behind-window-size	Specifies the size of the write-behind buffer for a single file or inode.	512 KB - 1 GB	1 MB
rebal-throttle	Rebalance process is made multithreaded to handle multiple files migration for enhancing the performance. During multiple file migration, there can be a severe impact on storage system performance. The throttling mechanism is provided to manage it.	lazy, normal, aggressive	normal

Option	Value Description	Allowed Values	Default Value
server.allow-insecure	Allows FUSE-based client connections from unprivileged ports. By default, this is enabled, meaning that ports can accept and reject messages from insecure ports. When disabled, only privileged ports are allowed. This is a global setting for allowing insecure ports to be enabled for all FUSE-based exports using a single option. Use **`nfs.rpc-auth-*`** options for NFS access control.	on | off	on
server.anongid	Value of the GID used for the anonymous user when root-squash is enabled. When root-squash is enabled, all the requests received from the root GID (that is 0) are changed to have the GID of the anonymous user.	0 - 4294967295	65534 (this UID is also known as nfsnobody)
server.anonuid	Value of the UID used for the anonymous user when root-squash is enabled. When root-squash is enabled, all the requests received from the root UID (that is 0) are changed to have the UID of the anonymous user.	0 - 4294967295	65534 (this UID is also known as nfsnobody)
server.event-threads	Specifies the number of network connections to be handled simultaneously by the server processes hosting a Red Hat Gluster Storage node.	1 - 32	1

Option	Value Description	Allowed Values	Default Value
server.gid-timeout	The time period in seconds which controls when cached groups has to expire. This is the cache that contains the groups (GIDs) where a specified user (UID) belongs to. This option is used only when **`server.manage-gids`** is enabled.	0-4294967295 seconds	2 seconds
server.manage-gids	Resolve groups on the server-side. By enabling this option, the groups (GIDs) a user (UID) belongs to gets resolved on the server, instead of using the groups that were send in the RPC Call by the client. This option makes it possible to apply permission checks for users that belong to bigger group lists than the protocol supports (approximately 93).	on|off	off
server.root-squash	Prevents root users from having root privileges, and instead assigns them the privileges of nfsnobody. This squashes the power of the root users, preventing unauthorized modification of files on the Red Hat Gluster Storage servers. This option is used only for glusterFS NFS protocol.	on | off	off
server.statedump-path	Specifies the directory in which the statedumpfiles must be stored.	/var/run/gluster (for a default installation)	Path to a directory
ssl.crl-path	Specifies the path to a file containing SSL certificate revocation list. This list helps the server nodes to stop the nodes with revoked certificates from accessing the cluster.	Path to a file in a directory	Path to a file

Option	Value Description	Allowed Values	Default Value
storage.health-check-interval	Sets the time interval in seconds for a filesystem health check. You can set it to 0 to disable. The POSIX translator on the bricks performs a periodic health check. If this check fails, the file system exported by the brick is not usable anymore and the brick process (glusterfsd) logs a warning and exits.	0-4294967295 seconds	30 seconds
storage.owner-gid	Sets the GID for the bricks of the volume. This option may be required when some of the applications need the brick to have a specific GID to function correctly. Example: For QEMU integration the UID/GID must be qemu:qemu, that is, 107:107 (107 is the UID and GID of qemu).	Any integer greater than or equal to -1.	The GID of the bricks are not changed. This is denoted by -1.
storage.owner-uid	Sets the UID for the bricks of the volume. This option may be required when some of the applications need the brick to have a specific UID to function correctly. Example: For QEMU integration the UID/GID must be **`qemu:qemu`**, that is, **`107:107`** (107 is the UID and GID of qemu).	Any integer greater than or equal to -1.	The UID of the bricks are not changed. This is denoted by -1.

Option	Value Description	Allowed Values	Default Value
storage.reserve	As of Red Hat Gluster Storage 3.4, the POSIX translator includes an option that allow users to reserve disk space on the bricks. This option ensures that enough space is retained to allow users to expand disks or cluster when the bricks are nearly full. The option does this by preventing new file creation when the disk has the storage.reserve percentage or less free space. storage.reserve option takes a percentage value, that saves up to the predetermined percentage of disk space. If set to **`0`** this option is disabled.	0-100	1 (1% of the brick size)
transport.listen-backlog	The maximum number of established TCP socket requests queued and waiting to be accepted at any one time.	0 to system maximum	10

11.4. Configuring a volume to be mounted read-only

Volumes can be mounted with read-only permissions at either the mount point or the volume level.

To specify that a volume can only be mounted read-only, enable the read-only volume option by running the following on any Red Hat Gluster Storage server that hosts that volume.

```
# gluster volume set volname read-only enable
```

11.5. Configuring Transport Types for a Volume

A volume can support one or more transport types for communication between clients and brick processes. There are three types of supported transport, which are, tcp, rdma, and tcp,rdma.

To change the supported transport types of a volume, follow the procedure:

1. Unmount the volume on all the clients using the following command:

   ```
   # umount mount-point
   ```

2. Stop the volumes using the following command:

```
# gluster volume stop volname
```

3. Change the transport type. For example, to enable both tcp and rdma execute the followimg command:

   ```
   # gluster volume set volname config.transport tcp,rdma OR tcp OR rdma
   ```

4. Mount the volume on all the clients. For example, to mount using rdma transport, use the following command:

   ```
   # mount -t glusterfs -o transport=rdma server1:/test-volume
   /mnt/glusterfs
   ```

11.6. Reserving Storage on a Volume

The POSIX translator is now enhanced with an option that allows user to reserve disk space on the bricks. Some administrative operations, like expanding storage or rebalancing data across nodes, require spare working space on the disk. The storage.reserve option lets users expand disk or cluster when backend bricks are full.

When the bricks have free space less than or equal to the reserved space, as declared by the user, new files are not created. This prevents ENOSPC errors on mount points.

To enable reserve option, execute the following command:

```
# gluster volume set volname storage.reserve percentage
```

Once this option is enabled, the reserved disk space is not used by the mount points. The **`storage.reserve`** option takes a percentage value. The default value for the option is 1(1% of the brick size). If set to 0 this option is disabled.

Note

When the disk space reduces to the size of reserved disk space, only internal operations like rebalance, self-heal and so on can be performed.

11.7. Expanding Volumes

Volumes can be expanded while the trusted storage pool is online and available. For example, you can add a brick to a distributed volume, which increases distribution and adds capacity to the Red Hat Gluster Storage volume. Similarly, you can add a group of bricks to a replicated or distributed replicated volume, which increases the capacity of the Red Hat Gluster Storage volume.

When expanding replicated or distributed replicated volumes, the number of bricks being added must be a multiple of the replica count. This also applies to arbitrated volumes. For example, to expand a distributed replicated volume with a replica count of 3, you need to add bricks in multiples of 3 (such as 6, 9, 12, etc.).

You can also convert a replica 2 volume into an arbitrated replica 3 volume by following the instructions in Section 5.8.5, “Converting to an arbitrated volume”.

Important

Converting an existing distribute volume to replicate or distribute-replicate volume is not supported.

Expanding a Volume

1. From any server in the trusted storage pool, use the following command to probe the server on which you want to add a new brick:

   ```
   # gluster peer probe HOSTNAME
   ```

   For example:

   ```
   # gluster peer probe server5
   Probe successful

   # gluster peer probe server6
   Probe successful
   ```

2. Add the bricks using the following command:

   ```
   # gluster volume add-brick VOLNAME NEW_BRICK
   ```

   For example:

   ```
   # gluster volume add-brick test-volume server5:/rhgs/brick5/
   server6:/rhgs/brick6/
   Add Brick successful
   ```

3. Check the volume information using the following command:

   ```
   # gluster volume info
   ```

   The command output displays information similar to the following:

   ```
   Volume Name: test-volume
   Type: Distribute-Replicate
   Status: Started
   Number of Bricks: 6
   Bricks:
   Brick1: server1:/rhgs/brick1
   Brick2: server2:/rhgs/brick2
   Brick3: server3:/rhgs/brick3
   Brick4: server4:/rhgs/brick4
   Brick5: server5:/rhgs/brick5
   Brick6: server6:/rhgs/brick6
   ```

4. Rebalance the volume to ensure that files will be distributed to the new brick. Use the rebalance command as described in Section 11.11, “Rebalancing Volumes”.

   The **add-brick** command should be followed by a **rebalance** operation to ensure better utilization of the added bricks.

11.7.1. Expanding a Tiered Volume

You can add a group of bricks to a cold tier volume and to the hot tier volume to increase the capacity of the Red Hat Gluster Storage volume.

11.7.1.1. Expanding a Cold Tier Volume

Expanding a cold tier volume is same as a non-tiered volume. If you are reusing the brick, ensure to perform the steps listed in "Section 5.4.3, " Reusing a Brick from a Deleted Volume "" section.

1. Detach the tier by performing the steps listed in Section 17.7, "Detaching a Tier from a Volume"

2. From any server in the trusted storage pool, use the following command to probe the server on which you want to add a new brick :

   ```
   # gluster peer probe HOSTNAME
   ```

   For example:

   ```
   # gluster peer probe server5
   Probe successful

   # gluster peer probe server6
   Probe successful
   ```

3. Add the bricks using the following command:

   ```
   # gluster volume add-brick VOLNAME NEW_BRICK
   ```

   For example:

   ```
   # gluster volume add-brick test-volume server5:/rhgs/brick5/
   server6:/rhgs/brick6/
   ```

4. Rebalance the volume to ensure that files will be distributed to the new brick. Use the rebalance command as described in Section 11.11, "Rebalancing Volumes".

   The **add-brick** command should be followed by a **rebalance** operation to ensure better utilization of the added bricks.

5. Reattach the tier to the volume with both old and new (expanded) bricks:

   # gluster volume tier *VOLNAME* attach [replica COUNT] NEW-BRICK...

   > **Important**
   >
   > When you reattach a tier, an internal process called fix-layout commences internally to prepare the hot tier for use. This process takes time and there will a delay in starting the tiering activities.

   If you are reusing the brick, be sure to clearly wipe the existing data before attaching it to the tiered volume.

11.7.1.2. Expanding a Hot Tier Volume

You can expand a hot tier volume by attaching and adding bricks for the hot tier.

1. Detach the tier by performing the steps listed in Section 17.7, “Detaching a Tier from a Volume”

2. Reattach the tier to the volume with both old and new (expanded) bricks:

   # gluster volume tier *VOLNAME* attach [replica COUNT] NEW-BRICK...

   For example,

   ```
   # gluster volume tier test-volume attach replica 2 server1:/rhgs/tier5
   server2:/rhgs/tier6 server1:/rhgs/tier7 server2:/rhgs/tier8
   ```

   Important

   When you reattach a tier, an internal process called fix-layout commences internally to prepare the hot tier for use. This process takes time and there will a delay in starting the tiering activities.

   If you are reusing the brick, be sure to clearly wipe the existing data before attaching it to the tiered volume.

11.7.2. Expanding a Dispersed or Distributed-dispersed Volume

Expansion of a dispersed or distributed-dispersed volume can be done by adding new bricks. The number of additional bricks should be in multiple of basic configuration of the volume. For example, if you have a volume with configuration (4+2 = 6), then you must only add 6 (4+2) or multiple of 6 bricks (such as 12, 18, 24 and so on).

Note

If you add bricks to a **`Dispersed`** volume, it will be converted to a**`Distributed-Dispersed`** volume, and the existing dispersed volume will be treated as dispersed subvolume.

1. From any server in the trusted storage pool, use the following command to probe the server on which you want to add new bricks:

   ```
   # gluster peer probe HOSTNAME
   ```

   For example:

   ```
   # gluster peer probe server4
   Probe successful

   # gluster peer probe server5
   Probe successful

   # gluster peer probe server6
   Probe successful
   ```

2. Add the bricks using the following command:

   ```
   # gluster volume add-brick VOLNAME NEW_BRICK
   ```

   For example:

   ```
   # gluster volume add-brick test-volume server4:/rhgs/brick7
   server4:/rhgs/brick8 server5:/rhgs/brick9 server5:/rhgs/brick10
   server6:/rhgs/brick11 server6:/rhgs/brick12
   ```

3. (Optional) View the volume information after adding the bricks:

   ```
   # gluster volume info VOLNAME
   ```

   For example:

   ```
   # gluster volume info test-volume
   Volume Name: test-volume
   Type: Distributed-Disperse
   Volume ID: 2be607f2-f961-4c4b-aa26-51dcb48b97df
   Status: Started
   Snapshot Count: 0
   Number of Bricks: 2 x (4 + 2) = 12
   Transport-type: tcp
   Bricks:
   Brick1: server1:/rhgs/brick1
   Brick2: server1:/rhgs/brick2
   Brick3: server2:/rhgs/brick3
   Brick4: server2:/rhgs/brick4
   Brick5: server3:/rhgs/brick5
   Brick6: server3:/rhgs/brick6
   Brick7: server4:/rhgs/brick7
   Brick8: server4:/rhgs/brick8
   Brick9: server5:/rhgs/brick9
   Brick10: server5:/rhgs/brick10
   Brick11: server6:/rhgs/brick11
   Brick12: server6:/rhgs/brick12
   Options Reconfigured:
   transport.address-family: inet
   performance.readdir-ahead: on
   nfs.disable: on
   ```

4. Rebalance the volume to ensure that the files will be distributed to the new brick. Use the rebalance command as described in Section 11.11, “Rebalancing Volumes”.

   The **add-brick** command should be followed by a **rebalance** operation to ensure better utilization of the added bricks.

11.8. Shrinking Volumes

You can shrink volumes while the trusted storage pool is online and available. For example, you may need to remove a brick that has become inaccessible in a distributed volume because of a hardware or network failure.

When shrinking distributed replicated volumes, the number of bricks being removed must be a multiple of the replica count. For example, to shrink a distributed replicated volume with a replica count of 3, you need to remove bricks in multiples of 3 (such as 6, 9, 12, etc.). In addition, the bricks you are removing must be from the same sub-volume (the same replica set). In a non-replicated volume, all bricks must be available in order to migrate data and perform the remove brick operation. In a replicated or arbitrated volume, at least one of the data bricks in the replica set must be available.

The guidelines are identical when removing a distribution set from a distributed replicated volume with arbiter bricks. If you want to reduce the replica count of an arbitrated distributed replicated volume to replica 2, you must remove only the arbiter bricks. If you want to reduce a volume from arbitrated distributed replicated to distributed only, remove the arbiter brick and one replica brick from each replica subvolume.

Shrinking a Volume

1. Remove a brick using the following command:

   ```
   # gluster volume remove-brick VOLNAME BRICK start
   ```

   For example:

   ```
   # gluster volume remove-brick test-volume server2:/rhgs/brick2 start
   Remove Brick start successful
   ```

   > **Note**
   >
   > If the **`remove-brick`** command is run with **`force`** or without any option, the data on the brick that you are removing will no longer be accessible at the glusterFS mount point. When using the **`start`** option, the data is migrated to other bricks, and on a successful commit the removed brick's information is deleted from the volume configuration. Data can still be accessed directly on the brick.

2. You can view the status of the remove brick operation using the following command:

   ```
   # gluster volume remove-brick VOLNAME BRICK status
   ```

   For example:

   ```
   # gluster volume remove-brick test-volume server2:/rhgs/brick2 status
         Node Rebalanced  size scanned failures skipped   status  run
   time
             -files                                          in h:m:s
   ----------  --------- ------ ------ -------- ------  --------- ----
   ----
   localhost        5032 43.4MB  27715       0    5604  completed
   0:15:05
   10.70.43.41         0 0Bytes      0       0       0  completed
   0:08:18

   volume rebalance: test-volume: success
   ```

3. When the data migration shown in the previous **`status`** command is complete, run the following command to commit the brick removal:

```
# gluster volume remove-brick VOLNAME BRICK commit
```

For example,

```
# gluster volume remove-brick test-volume server2:/rhgs/brick2 commit
```

4. After the brick removal, you can check the volume information using the following command:

```
# gluster volume info
```

The command displays information similar to the following:

```
# gluster volume info
Volume Name: test-volume
Type: Distribute
Status: Started
Number of Bricks: 3
Bricks:
Brick1: server1:/rhgs/brick1
Brick3: server3:/rhgs/brick3
Brick4: server4:/rhgs/brick4
```

11.8.1. Shrinking a Geo-replicated Volume

1. Remove a brick using the following command:

```
# gluster volume remove-brick VOLNAME BRICK start
```

For example:

```
# gluster volume remove-brick MASTER_VOL MASTER_HOST:/rhgs/brick2
start
Remove Brick start successful
```

> **Note**
>
> If the **remove-brick** command is run with **force** or without any option, the data on the brick that you are removing will no longer be accessible at the glusterFS mount point. When using the **start** option, the data is migrated to other bricks, and on a successful commit the removed brick's information is deleted from the volume configuration. Data can still be accessed directly on the brick.

2. Use geo-replication **config checkpoint** to ensure that all the data in that brick is synced to the slave.

   a. Set a checkpoint to help verify the status of the data synchronization.

   ```
   # gluster volume geo-replication MASTER_VOL
   SLAVE_HOST::SLAVE_VOL config checkpoint now
   ```

b. Verify the checkpoint completion for the geo-replication session using the following command:

```
# gluster volume geo-replication MASTER_VOL
SLAVE_HOST::SLAVE_VOL status detail
```

3. You can view the status of the remove brick operation using the following command:

   ```
   # gluster volume remove-brick VOLNAME BRICK status
   ```

   For example:

   ```
   # gluster volume remove-brick  MASTER_VOL MASTER_HOST:/rhgs/brick2
   status
   ```

4. Stop the geo-replication session between the master and the slave:

   ```
   # gluster volume geo-replication MASTER_VOL SLAVE_HOST::SLAVE_VOL stop
   ```

5. When the data migration shown in the previous **`status`** command is complete, run the following command to commit the brick removal:

   ```
   # gluster volume remove-brick VOLNAME BRICK commit
   ```

   For example,

   ```
   # gluster volume remove-brick  MASTER_VOL MASTER_HOST:/rhgs/brick2
   commit
   ```

6. After the brick removal, you can check the volume information using the following command:

   ```
   # gluster volume info
   ```

7. Start the geo-replication session between the hosts:

   ```
   # gluster volume geo-replication MASTER_VOL SLAVE_HOST::SLAVE_VOL
   start
   ```

11.8.2. Shrinking a Tiered Volume

You can shrink a tiered volume while the trusted storage pool is online and available. For example, you may need to remove a brick that has become inaccessible because of a hardware or network failure.

11.8.2.1. Shrinking a Cold Tier Volume

1. Detach the tier by performing the steps listed in Section 17.7, “Detaching a Tier from a Volume”

2. Remove a brick using the following command:

   ```
   # gluster volume remove-brick VOLNAME BRICK start
   ```

   For example:

```
# gluster volume remove-brick test-volume server2:/rhgs/brick2 start
Remove Brick start successful
```

> **Note**
>
> If the **remove-brick** command is run with **force** or without any option, the data on the brick that you are removing will no longer be accessible at the glusterFS mount point. When using the **start** option, the data is migrated to other bricks, and on a successful commit the removed brick's information is deleted from the volume configuration. Data can still be accessed directly on the brick.

3. You can view the status of the remove brick operation using the following command:

   ```
   # gluster volume remove-brick VOLNAME BRICK status
   ```

   For example:

   ```
   # gluster volume remove-brick test-volume server2:/rhgs/brick2 status
         Node    Rebalanced-files          size       scanned
   failures         status
    ---------         -----------   -----------   -----------   ------
   -----   ------------
    localhost                    16      16777216            52
   0    in progress
   192.168.1.1                  13      16723211            47
   0    in progress
   ```

4. When the data migration shown in the previous **status** command is complete, run the following command to commit the brick removal:

   ```
   # gluster volume remove-brick VOLNAME BRICK commit
   ```

   For example,

   ```
   # gluster volume remove-brick test-volume server2:/rhgs/brick2 commit
   ```

5. Rerun the attach-tier command only with the required set of bricks:

   # gluster volume tier *VOLNAME* attach [replica *COUNT*] *BRICK*...

   For example,

   ```
   # gluster volume tier test-volume attach replica 2 server1:/rhgs/tier1
   server2:/rhgs/tier2 server1:/rhgs/tier3 server2:/rhgs/tier4
   ```

Important

When you attach a tier, an internal process called fix-layout commences internally to prepare the hot tier for use. This process takes time and there will a delay in starting the tiering activities.

11.8.2.2. Shrinking a Hot Tier Volume

You must first decide on which bricks should be part of the hot tiered volume and which bricks should be removed from the hot tier volume.

1. Detach the tier by performing the steps listed in Section 17.7, “Detaching a Tier from a Volume”

2. Rerun the attach-tier command only with the required set of bricks:

   ```
   # gluster volume tier VOLNAME attach [replica COUNT] brick...
   ```

Important

When you reattach a tier, an internal process called fix-layout commences internally to prepare the hot tier for use. This process takes time and there will a delay in starting the tiering activities.

11.8.3. Stopping a `remove-brick` Operation

A **`remove-brick`** operation that is in progress can be stopped by using the **`stop`** command.

Note

Files that were already migrated during **`remove-brick`** operation will not be migrated back to the same brick when the operation is stopped.

To stop remove brick operation, use the following command:

```
# gluster volume remove-brick VOLNAME BRICK stop
```

For example:

```
gluster volume remove-brick test-volume server1:/rhgs/brick1/
server2:/brick2/ stop

Node   Rebalanced-files   size     scanned  failures   skipped   status
run-time in secs
----      -------         ----       ----     ------    -----     -----
------
localhost     23          376Bytes    34        0        0      stopped
2.00
rhs1          0           0Bytes      88        0        0      stopped
2.00
```

```
rhs2           0            0Bytes       0         0         0       not
started  0.00
'remove-brick' process may be in the middle of a file migration.
The process will be fully stopped once the migration of the file is
complete.
Please check remove-brick process for completion before doing any further
brick related tasks on the volume.
```

11.9. Migrating Volumes

Data can be redistributed across bricks while the trusted storage pool is online and available.Before replacing bricks on the new servers, ensure that the new servers are successfully added to the trusted storage pool.

Note

Before performing a **replace-brick** operation, review the known issues related to **replace-brick** operation in the Red Hat Gluster Storage Release Notes.

11.9.1. Replacing a Subvolume on a Distribute or Distribute-replicate Volume

This procedure applies only when at least one brick from the subvolume to be replaced is online. In case of a Distribute volume, the brick that must be replaced must be online. In case of a Distribute-replicate, at least one brick from the subvolume from the replica set that must be replaced must be online.

To replace the entire subvolume with new bricks on a *Distribute-replicate* volume, follow these steps:

1. Add the new bricks to the volume.

   ```
   # gluster volume add-brick VOLNAME [replica <COUNT>] NEW-BRICK
   ```

   Example 11.1. Adding a Brick to a Distribute Volume

   ```
   # gluster volume add-brick test-volume server5:/rhgs/brick5
   Add Brick successful
   ```

2. Verify the volume information using the command:

   ```
   # gluster volume info
    Volume Name: test-volume
       Type: Distribute
       Status: Started
       Number of Bricks: 5
       Bricks:
       Brick1: server1:/rhgs/brick1
       Brick2: server2:/rhgs/brick2
       Brick3: server3:/rhgs/brick3
       Brick4: server4:/rhgs/brick4
       Brick5: server5:/rhgs/brick5
   ```

> **Note**
>
> In case of a Distribute-replicate volume, you must specify the replica count in the **add-brick** command and provide the same number of bricks as the replica count to the **add-brick** command.

3. Remove the bricks to be replaced from the subvolume.

   a. Start the **remove-brick** operation using the command:

```
# gluster volume remove-brick VOLNAME [replica <COUNT>] <BRICK>
start
```

Example 11.2. Start a remove-brick operation on a distribute volume

```
# gluster volume remove-brick test-volume server2:/rhgs/brick2
start
Remove Brick start successful
```

   b. View the status of the **remove-brick** operation using the command:

```
# gluster volume remove-brick VOLNAME [replica <COUNT>] BRICK
status
```

Example 11.3. View the Status of remove-brick Operation

```
# gluster volume remove-brick test-volume server2:/rhgs/brick2
status

Node   Rebalanced-files size   scanned  failures  skipped
status  run-time in

h:m:s
----      -------       ----    ----     ------    -----
-----     ------
server2   10045       204.9MB   73522      0         0
in progress  0:10:34

Estimated time left for rebalance to complete: 0:10:23
```

Keep monitoring the **remove-brick** operation status by executing the above command. In the above example, the estimated time for rebalance to complete is 10 minutes. When the value of the status field is set to **complete** in the output of **remove-brick** status command, proceed further.

   c. Commit the **remove-brick** operation using the command:

```
# gluster volume remove-brick VOLNAME [replica <COUNT>] <BRICK>
commit
```

Example 11.4. Commit the remove-brick Operation on a Distribute Volume

```
# gluster volume remove-brick test-volume server2:/rhgs/brick2
commit
```

d. Verify the volume information using the command:

```
# gluster volume info
Volume Name: test-volume
Type: Distribute
Status: Started
Number of Bricks: 4
Bricks:
Brick1: server1:/rhgs/brick1
Brick3: server3:/rhgs/brick3
Brick4: server4:/rhgs/brick4
Brick5: server5:/rhgs/brick5
```

e. Verify the content on the brick after committing the **`remove-brick`** operation on the volume. If there are any files leftover, copy it through FUSE or NFS mount.

   a. Verify if there are any pending files on the bricks of the subvolume.

      Along with files, all the application-specific extended attributes must be copied. glusterFS also uses extended attributes to store its internal data. The extended attributes used by glusterFS are of the form **`trusted.glusterfs.*`**, **`trusted.afr.*`**, and **`trusted.gfid`**. Any extended attributes other than ones listed above must also be copied.

      To copy the application-specific extended attributes and to achieve a an effect similar to the one that is described above, use the following shell script:

      Syntax:

      ```
      # copy.sh <glusterfs-mount-point> <brick>
      ```

      Example 11.5. Code Snippet Usage

      If the mount point is **`/mnt/glusterfs`** and brick path is **`/rhgs/brick1`**, then the script must be run as:

      ```
      # copy.sh /mnt/glusterfs /rhgs/brick1
      ```

      ```
      #!/bin/bash

      MOUNT=$1
      BRICK=$2

      for file in `find $BRICK ! -type d`; do
          rpath=`echo $file | sed -e "s#$BRICK\(.*\)#\1#g"`
      ```

```
    rdir=`dirname $rpath`

    cp -fv $file $MOUNT/$rdir;

    for xattr in `getfattr -e hex -m. -d $file 2>/dev/null
| sed -e '/^#/d' | grep -v -E "trusted.glusterfs.*" | grep
-v -E "trusted.afr.*" | grep -v "trusted.gfid"`;
    do
        key=`echo $xattr | cut -d"=" -f 1`
        value=`echo $xattr | cut -d"=" -f 2`

        setfattr $MOUNT/$rpath -n $key -v $value
    done
done
```

b. To identify a list of files that are in a split-brain state, execute the command:

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

c. If there are any files listed in the output of the above command, compare the files across the bricks in a replica set, delete the bad files from the brick and retain the correct copy of the file. Manual intervention by the System Administrator would be required to choose the correct copy of file.

11.9.2. Replacing an Old Brick with a New Brick on a Replicate or Distribute-replicate Volume

A single brick can be replaced during a hardware failure situation, such as a disk failure or a server failure. The brick that must be replaced could either be online or offline. This procedure is applicable for volumes with replication. In case of a *Replicate* or *Distribute-replicate* volume types, after replacing the brick, self-heal is automatically triggered to heal the data on the new brick.

Procedure to replace an old brick with a new brick on a *Replicate* or *Distribute-replicate* volume:

1. Ensure that the new brick (**server5:/rhgs/brick1**) that replaces the old brick (**server0:/rhgs/brick1**) is empty. Ensure that all the bricks are online. The brick that must be replaced can be in an offline state.

2. Execute the **replace-brick** command with the **force** option:

```
# gluster volume replace-brick test-volume server0:/rhgs/brick1
server5:/rhgs/brick1 commit force
volume replace-brick: success: replace-brick commit successful
```

3. Check if the new brick is online.

```
# gluster volume status
Status of volume: test-volume
Gluster process                          Port    Online    Pid
---------------------------------------------------------
Brick server5:/rhgs/brick1                  49156     Y     5731

Brick server1:/rhgs/brick1                  49153     Y     5354
Brick server2:/rhgs/brick1            49154    Y    5365

Brick server3:/rhgs/brick1            49155    Y    5376
```

4. Data on the newly added brick would automatically be healed. It might take time depending upon the amount of data to be healed. It is recommended to check heal information after replacing a brick to make sure all the data has been healed before replacing/removing any other brick.

   ```
   # gluster volume heal VOL_NAME info
   ```

   For example:

   ```
   # gluster volume heal test-volume info
   Brick server5:/rhgs/brick1
   Status: Connected
   Number of entries: 0

   Brick server1:/rhgs/brick1
   Status: Connected
   Number of entries: 0

   Brick server2:/rhgs/brick1
   Status: Connected
   Number of entries: 0

   Brick server3:/rhgs/brick1
   Status: Connected
   Number of entries: 0
   ```

   The value of **`Number of entries`** field will be displayed as zero if the heal is complete.

11.9.3. Replacing an Old Brick with a New Brick on a Distribute Volume

1. Before making any changes, check the contents of the brick that you want to remove from the volume.

   ```
   # ls /mount/point/OLDBRICK
   file1
   file2
   ...
   file5
   ```

2. Add the new brick to the volume.

   ```
   # gluster volume add-brick VOLNAME NEWSERVER:NEWBRICK
   ```

3. Start removing the old brick.

   ```
   # gluster volume remove-brick VOLNAME OLDSERVER:OLDBRICK start
   ```

4. Wait until the remove-brick status command shows that the removal is complete.

```
# gluster volume remove-brick VOLNAME BRICK status
          Rebalanced
run time
Node      files       size    scanned   failures    skipped   status
in secs
------------------------------------------------------------------
--------------
localhost 5           20Bytes 15        0           0
completed 0.00
```

5. Finish removing the old brick.

```
# gluster volume remove-brick VOLNAME OLDSERVER:OLDBRICK commit
```

6. Verify that all files that were on the removed brick are still present on the volume.

11.9.4. Replacing an Old Brick with a New Brick on a Dispersed or Distributed-dispersed Volume

A single brick can be replaced during a hardware failure situation, such as a disk failure or a server failure. The brick that must be replaced could either be online or offline but all other bricks must be online.

Procedure to replace an old brick with a new brick on a Dispersed or Distributed-dispersed volume:

1. Ensure that the new brick that replaces the old brick is empty. The brick that must be replaced can be in an offline state but all other bricks must be online.

2. Execute the replace-brick command with the **force** option:

```
# gluster volume replace-brick VOL_NAME old_brick_path new_brick_path
commit force
```

For example:

```
# gluster volume replace-brick test-volume server1:/rhgs/brick2
server1:/rhgs/brick2new  commit force
volume replace-brick: success: replace-brick commit successful
```

The new brick you are adding could be from the same server or you can add a new server and then a new brick.

3. Check if the new brick is online.

```
# gluster volume status
Status of volume: test-volume
Gluster process                  TCP Port  RDMA Port  Online    Pid
------------------------------------------------------------------
------------
Brick server1:/rhgs/brick1       49187     0          Y       19927
Brick server1:/rhgs/brick2new    49188     0          Y       19946
Brick server2:/rhgs/brick3       49189     0          Y       19965
Brick server2:/rhgs/brick4       49190     0          Y       19984
Brick server3:/rhgs/brick5       49191     0          Y       20003
Brick server3:/rhgs/brick6       49192     0          Y       20022
```

```
NFS Server on localhost              N/A       N/A        N          N/A
Self-heal Daemon on localhost        N/A       N/A        Y
20043

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

4. Data on the newly added brick would automatically be healed. It might take time depending upon the amount of data to be healed. It is recommended to check heal information after replacing a brick to make sure all the data has been healed before replacing/removing any other brick.

   ```
   # gluster volume heal VOL_NAME info
   ```

   For example:

   ```
   # gluster volume heal test-volume info
   Brick server1:/rhgs/brick1
   Status: Connected
   Number of entries: 0

   Brick server1:/rhgs/brick2new
   Status: Connected
   Number of entries: 0

   Brick server2:/rhgs/brick3
   Status: Connected
   Number of entries: 0

   Brick server2:/rhgs/brick4
   Status: Connected
   Number of entries: 0

   Brick server3:/rhgs/brick5
   Status: Connected
   Number of entries: 0

   Brick server3:/rhgs/brick6
   Status: Connected
   Number of entries: 0
   ```

   The value of **`Number of entries`** field will be displayed as zero if the heal is complete.

5. Red Hat Gluster Storage 3.4 introduces the ‘summary’ option of ‘heal info’ command. This command displays the statistics of entries pending heal in split-brain and the entries undergoing healing. This command prints only the entry count and not the actual file-names or gfids.

   To get the summary of a volume, run the following command:

   ```
   # gluster volume heal VOLNAME info summary
   ```

   For example:

   ```
   # gluster volume heal test-volume info summary
   ```

```
            Command output: Brick 192.168.2.8:/brick/1
            Status: Connected
            Total Number of entries: 363
            Number of entries in heal pending: 362
            Number of entries in split-brain: 0
            Number of entries possibly healing: 1
```

> **Note**
>
> The ‘summary’ option provides a detailed information about the brick unlike the ‘info’ command. The summary information is obtained in a similar way as the ‘info’ command.

The --xml parameter provides the output of the summary option in XML format

```
# gluster volume heal test-volume info summary --xml

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
  <cliOutput>
    <healInfo<
      <bricks>
        <brick hostUuid="9105dd4b-eca8-4fdb-85b2-b81cdf77eda3">
          <name>192.168.2.8:/brick/1</name>
            <status>Connected</status>
              <totalNumberOfEntries>363</totalNumberOfEntries>

<numberOfEntriesInHealPending>362</numberOfEntriesInHealPending>

<numberOfEntriesInSplitBrain>0</numberOfEntriesInSplitBrain>

<numberOfEntriesPossiblyHealing>1</numberOfEntriesPossiblyHealing>
            </brick>
          </bricks>
        </healInfo>
      <opRet>0</opRet>
    <opErrno>0</opErrno>
  <opErrstr/>
</cliOutput>
```

11.9.5. Reconfiguring a Brick in a Volume

The **`reset-brick`** subcommand is useful when you want to reconfigure a brick rather than replace it. **`reset-brick`** lets you replace a brick with another brick of the same location and UUID. For example, if you initially configured bricks so that they were identified with a hostname, but you want to use that hostname somewhere else, you can use **`reset-brick`** to stop the brick, reconfigure it so that it is identified by an IP address instead of the hostname, and return the reconfigured brick to the cluster.

To reconfigure a brick (replace a brick with another brick of the same hostname, path, and UUID), perform the following steps:

1. Ensure that the quorum minimum will still be met when the brick that you want to reset is taken offline

2. If possible, Red Hat recommends stopping I/O, and verifying that no heal operations are pending on the volume.

3. Run the following command to kill the brick that you want to reset.

   ```
   # gluster volume reset-brick VOLNAME HOSTNAME:BRICKPATH start
   ```

4. Configure the offline brick according to your needs.

5. Check that the volume's **Volume ID** displayed by **gluster volume info** matches the **volume-id** (if any) of the offline brick.

   ```
   # gluster volume info VOLNAME
   # cat /var/lib/glusterd/vols/VOLNAME/VOLNAME.HOSTNAME.BRICKPATH.vol |
   grep volume-id
   ```

   For example, in the following dispersed volume, the **Volume ID** and the **volume-id** are both **ab8a981a-a6d9-42f2-b8a5-0b28fe2c4548**.

   ```
   # gluster volume info vol
   Volume Name: vol
   Type: Disperse
   Volume ID: ab8a981a-a6d9-42f2-b8a5-0b28fe2c4548
   Status: Started
   Snapshot Count: 0
   Number of Bricks: 1 x (4 + 2) = 6
   Transport-type: tcp
   Bricks:
   Brick1: myhost:/brick/gluster/vol-1
   ```

   ```
   # cat /var/lib/glusterd/vols/vol/vol.myhost.brick-gluster-vol-1.vol |
   grep volume-id
   option volume-id ab8a981a-a6d9-42f2-b8a5-0b28fe2c4548
   ```

6. Bring the reconfigured brick back online. There are two options for this:

   - If your brick did not have a ***volume-id*** in the previous step, run:

     ```
     # gluster volume reset-brick VOLNAME HOSTNAME:BRICKPATH
     HOSTNAME:BRICKPATH commit
     ```

   - If your brick's ***volume-id*** matches your volume's identifier, Red Hat recommends adding the **force** keyword to ensure that the operation succeeds.

     ```
     # gluster volume reset-brick VOLNAME HOSTNAME:BRICKPATH
     HOSTNAME:BRICKPATH commit force
     ```

11.10. Replacing Hosts

11.10.1. Replacing a Host Machine with a Different Hostname

You can replace a failed host machine with another host that has a different hostname.

Important

Ensure that the new peer has the exact disk capacity as that of the one it is replacing. For example, if the peer in the cluster has two 100GB drives, then the new peer must have the same disk capacity and number of drives.

In the following example the original machine which has had an irrecoverable failure is **`server0.example.com`** and the replacement machine is **`server5.example.com`**. The brick with an unrecoverable failure is **`server0.example.com:/rhgs/brick1`** and the replacement brick is **`server5.example.com:/rhgs/brick1`**.

1. Stop the geo-replication session if configured by executing the following command:

   ```
   # gluster volume geo-replication MASTER_VOL SLAVE_HOST::SLAVE_VOL stop
   force
   ```

2. Probe the new peer from one of the existing peers to bring it into the cluster.

   ```
   # gluster peer probe server5.example.com
   ```

3. Ensure that the new brick **`(server5.example.com:/rhgs/brick1)`** that is replacing the old brick **`(server0.example.com:/rhgs/brick1)`** is empty.

4. If the geo-replication session is configured, perform the following steps:

   a. Setup the geo-replication session by generating the ssh keys:

      ```
      # gluster system:: execute gsec_create
      ```

   b. Create geo-replication session again with **`force`** option to distribute the keys from new nodes to Slave nodes.

      ```
      # gluster volume geo-replication MASTER_VOL
      SLAVE_HOST::SLAVE_VOL create push-pem force
      ```

   c. After successfully setting up the shared storage volume, when a new node is replaced in the cluster, the shared storage is not mounted automatically on this node. Neither is the **`/etc/fstab`** entry added for the shared storage on this node. To make use of shared storage on this node, execute the following commands:

      ```
      # mount -t glusterfs local node's ip:gluster_shared_storage
      /var/run/gluster/shared_storage
      # cp /etc/fstab /var/run/gluster/fstab.tmp
      # echo  local node's ip:/gluster_shared_storage
      /var/run/gluster/shared_storage/ glusterfs defaults 0 0" >>
      /etc/fstab
      ```

      For more information on setting up shared storage volume, see Section 11.12, “Setting up Shared Storage Volume”.

   d. Configure the meta-volume for geo-replication:

```
# gluster volume geo-replication MASTER_VOL
SLAVE_HOST::SLAVE_VOL config use_meta_volume true
```

For more information on configuring meta-volume, see Section 10.3.5, “Configuring a Meta-Volume”.

5. Retrieve the brick paths in **server0.example.com** using the following command:

```
# gluster volume info <VOLNAME>
```

```
Volume Name: vol
Type: Replicate
Volume ID: 0xde822e25ebd049ea83bfaa3c4be2b440
Status: Started
Snap Volume: no
Number of Bricks: 1 x 2 = 2
Transport-type: tcp
Bricks:
Brick1: server0.example.com:/rhgs/brick1
Brick2: server1.example.com:/rhgs/brick1
Options Reconfigured:
performance.readdir-ahead: on
snap-max-hard-limit: 256
snap-max-soft-limit: 90
auto-delete: disable
```

Brick path in **server0.example.com** is **/rhgs/brick1**. This has to be replaced with the brick in the newly added host, **server5.example.com**.

6. Create the required brick path in server5.example.com.For example, if /rhs/brick is the XFS mount point in server5.example.com, then create a brick directory in that path.

```
# mkdir /rhgs/brick1
```

7. Execute the **replace-brick** command with the force option:

```
# gluster volume replace-brick vol server0.example.com:/rhgs/brick1
server5.example.com:/rhgs/brick1 commit force
volume replace-brick: success: replace-brick commit successful
```

8. Verify that the new brick is online.

```
# gluster volume status
Status of volume: vol
Gluster process                                  Port    Online Pid
Brick server5.example.com:/rhgs/brick1           49156    Y     5731
Brick server1.example.com:/rhgs/brick1            49153    Y     5354
```

9. Initiate self-heal on the volume. The status of the heal process can be seen by executing the command:

```
# gluster volume heal VOLNAME
```

10. The status of the heal process can be seen by executing the command:

    ```
    # gluster volume heal VOLNAME info
    ```

11. Detach the original machine from the trusted pool.

    ```
    # gluster peer detach server0.example.com
    ```

12. Ensure that after the self-heal completes, the extended attributes are set to zero on the other bricks in the replica.

    ```
    # getfattr -d -m. -e hex /rhgs/brick1
    getfattr: Removing leading '/' from absolute path names
    #file: rhgs/brick1
    security.selinux=0x756e636f6e66696e65645f753a6f626a6563745f723a66696c6
    55f743a733000
    trusted.afr.vol-client-0=0x000000000000000000000000
    trusted.afr.vol-client-1=0x000000000000000000000000
    trusted.gfid=0x00000000000000000000000000000001
    trusted.glusterfs.dht=0x0000000100000000000000007ffffffe
    trusted.glusterfs.volume-id=0xde822e25ebd049ea83bfaa3c4be2b440
    ```

    In this example, the extended attributes **`trusted.afr.vol-client-0`** and **`trusted.afr.vol-client-1`** have zero values. This means that the data on the two bricks is identical. If these attributes are not zero after self-heal is completed, the data has not been synchronised correctly.

13. Start the geo-replication session using **`force`** option:

    ```
    # gluster volume geo-replication MASTER_VOL SLAVE_HOST::SLAVE_VOL
    start force
    ```

11.10.2. Replacing a Host Machine with the Same Hostname

You can replace a failed host with another node having the same FQDN (Fully Qualified Domain Name). A host in a Red Hat Gluster Storage Trusted Storage Pool has its own identity called the UUID generated by the glusterFS Management Daemon.The UUID for the host is available in **`/var/lib/glusterd/glusterd/info`** file.

In the following example, the host with the FQDN as server0.example.com was irrecoverable and must to be replaced with a host, having the same FQDN. The following steps have to be performed on the new host.

1. Stop the geo-replication session if configured by executing the following command:

    ```
     # gluster volume geo-replication MASTER_VOL SLAVE_HOST::SLAVE_VOL
    stop force
    ```

2. Stop the **`glusterd`** service on the server0.example.com.

    ```
    # service glusterd stop
    ```

Important

If **glusterd** crashes, there is no functionality impact to this crash as it occurs during the shutdown. For more information, see Section 24.3, “Resolving **glusterd** Crash”

3. Retrieve the UUID of the failed host (server0.example.com) from another of the Red Hat Gluster Storage Trusted Storage Pool by executing the following command:

```
# gluster peer status
Number of Peers: 2

Hostname: server1.example.com
Uuid: 1d9677dc-6159-405e-9319-ad85ec030880
State: Peer in Cluster (Connected)

Hostname: server0.example.com
Uuid: b5ab2ec3-5411-45fa-a30f-43bd04caf96b
State: Peer Rejected (Connected)
```

   Note that the UUID of the failed host is **b5ab2ec3-5411-45fa-a30f-43bd04caf96b**

4. Edit the **glusterd.info** file in the new host and include the UUID of the host you retrieved in the previous step.

```
# cat /var/lib/glusterd/glusterd.info
UUID=b5ab2ec3-5411-45fa-a30f-43bd04caf96b
operating-version=30703
```

Note

The operating version of this node must be same as in other nodes of the trusted storage pool.

5. Select any host (say for example, server1.example.com) in the Red Hat Gluster Storage Trusted Storage Pool and retrieve its UUID from the **glusterd.info** file.

```
# grep -i uuid /var/lib/glusterd/glusterd.info
UUID=8cc6377d-0153-4540-b965-a4015494461c
```

6. Gather the peer information files from the host (server1.example.com) in the previous step. Execute the following command in that host (server1.example.com) of the cluster.

```
# cp -a /var/lib/glusterd/peers /tmp/
```

7. Remove the peer file corresponding to the failed host (server0.example.com) from the **/tmp/peers** directory.

```
# rm /tmp/peers/b5ab2ec3-5411-45fa-a30f-43bd04caf96b
```

Note that the UUID corresponds to the UUID of the failed host (server0.example.com) retrieved in Step 3.

8. Archive all the files and copy those to the failed host(server0.example.com).

   ```
   # cd /tmp; tar -cvf peers.tar peers
   ```

9. Copy the above created file to the new peer.

   ```
   # scp /tmp/peers.tar root@server0.example.com:/tmp
   ```

10. Copy the extracted content to the **/var/lib/glusterd/peers** directory. Execute the following command in the newly added host with the same name (server0.example.com) and IP Address.

    ```
    # tar -xvf /tmp/peers.tar
    # cp peers/* /var/lib/glusterd/peers/
    ```

11. Select any other host in the cluster other than the node (server1.example.com) selected in step 5. Copy the peer file corresponding to the UUID of the host retrieved in Step 5 to the new host (server0.example.com) by executing the following command:

    ```
    # scp /var/lib/glusterd/peers/<UUID-retrieved-from-step5>
    root@Example1:/var/lib/glusterd/peers/
    ```

12. Retrieve the brick directory information, by executing the following command in any host in the cluster.

    ```
    # gluster volume info
    Volume Name: vol
    Type: Replicate
    Volume ID: 0x8f16258c88a0498fbd53368706af7496
    Status: Started
    Snap Volume: no
    Number of Bricks: 1 x 2 = 2
    Transport-type: tcp
    Bricks:
    Brick1: server0.example.com:/rhgs/brick1
    Brick2: server1.example.com:/rhgs/brick1
    Options Reconfigured:
    performance.readdir-ahead: on
    snap-max-hard-limit: 256
    snap-max-soft-limit: 90
    auto-delete: disable
    ```

    In the above example, the brick path in server0.example.com is, **/rhgs/brick1**. If the brick path does not exist in server0.example.com, perform steps a, b, and c.

    a. Create a brick path in the host, server0.example.com.

       ```
       mkdir /rhgs/brick1
       ```

    b. Retrieve the volume ID from the existing brick of another host by executing the following command on any host that contains the bricks for the volume.

```
# getfattr -d -m. -ehex <brick-path>
```

Copy the volume-id.

```
# getfattr -d -m. -ehex /rhgs/brick1
getfattr: Removing leading '/' from absolute path names
# file: rhgs/brick1
trusted.afr.vol-client-0=0x000000000000000000000000
trusted.afr.vol-client-1=0x000000000000000000000000
trusted.gfid=0x00000000000000000000000000000001
trusted.glusterfs.dht=0x0000000100000000000000007ffffffe
trusted.glusterfs.volume-id=0x8f16258c88a0498fbd53368706af7496
```

In the above example, the volume id is 0x8f16258c88a0498fbd53368706af7496

c. Set this volume ID on the brick created in the newly added host and execute the following command on the newly added host (server0.example.com).

```
# setfattr -n trusted.glusterfs.volume-id -v <volume-id> <brick-
path>
```

For Example:

```
# setfattr -n trusted.glusterfs.volume-id -v
0x8f16258c88a0498fbd53368706af7496 /rhs/brick2/drv2
```

Data recovery is possible only if the volume type is replicate or distribute-replicate. If the volume type is plain distribute, you can skip steps 12 and 13.

13. Create a FUSE mount point to mount the glusterFS volume.

```
# mount -t glusterfs <server-name>:/VOLNAME <mount>
```

14. Perform the following operations to change the Automatic File Replication extended attributes so that the heal process happens from the other brick (server1.example.com:/rhgs/brick1) in the replica pair to the new brick (server0.example.com:/rhgs/brick1). Note that /mnt/r2 is the FUSE mount path.

a. Create a new directory on the mount point and ensure that a directory with such a name is not already present.

```
# mkdir /mnt/r2/<name-of-nonexistent-dir>
```

b. Delete the directory and set the extended attributes.

```
# rmdir /mnt/r2/<name-of-nonexistent-dir>
# setfattr -n trusted.non-existent-key -v abc /mnt/r2
# setfattr -x trusted.non-existent-key /mnt/r2
```

c. Ensure that the extended attributes on the other bricks in the replica (in this example, **trusted.afr.vol-client-0**) is not set to zero.

```
# getfattr -d -m. -e hex /rhgs/brick1 # file: rhgs/brick1
security.selinux=0x756e636f6e66696e65645f753a6f626a6563745f723a6
```

```
6696c655f743a733000
trusted.afr.vol-client-0=0x000000000000000300000002
trusted.afr.vol-client-1=0x000000000000000000000000
trusted.gfid=0x00000000000000000000000000000001
trusted.glusterfs.dht=0x0000000100000000000000007ffffffe
trusted.glusterfs.volume-id=0x8f16258c88a0498fbd53368706af7496
```

> **Note**
>
> You must ensure to perform steps 12, 13, and 14 for all the volumes having bricks from **server0.example.com**.

15. Start the **glusterd** service.

    ```
    # service glusterd start
    ```

16. Perform the self-heal operation on the restored volume.

    ```
    # gluster volume heal VOLNAME
    ```

17. You can view the gluster volume self-heal status by executing the following command:

    ```
    # gluster volume heal VOLNAME info
    ```

18. If the geo-replication session is configured, perform the following steps:

    a. Setup the geo-replication session by generating the ssh keys:

    ```
    # gluster system:: execute gsec_create
    ```

    b. Create geo-replication session again with **force** option to distribute the keys from new nodes to Slave nodes.

    ```
    # gluster volume geo-replication MASTER_VOL
    SLAVE_HOST::SLAVE_VOL create push-pem force
    ```

    c. After successfully setting up the shared storage volume, when a new node is replaced in the cluster, the shared storage is not mounted automatically on this node. Neither is the **/etc/fstab** entry added for the shared storage on this node. To make use of shared storage on this node, execute the following commands:

    ```
    # mount -t glusterfs <local node's ip>:gluster_shared_storage
    /var/run/gluster/shared_storage # cp /etc/fstab
    /var/run/gluster/fstab.tmp # echo "<local node's
    ip>:/gluster_shared_storage /var/run/gluster/shared_storage/
    glusterfs defaults 0 0" >> /etc/fstab
    ```

    For more information on setting up shared storage volume, see Section 11.12, “Setting up Shared Storage Volume”.

    d. Configure the meta-volume for geo-replication:

```
# gluster volume geo-replication MASTER_VOL
SLAVE_HOST::SLAVE_VOL config use_meta_volume true
```

e. Start the geo-replication session using **force** option:

```
# gluster volume geo-replication MASTER_VOL
SLAVE_HOST::SLAVE_VOL start force
```

Replacing a host with the same Hostname in a two-node Red Hat Gluster Storage Trusted Storage Pool

If there are only 2 hosts in the Red Hat Gluster Storage Trusted Storage Pool where the host server0.example.com must be replaced, perform the following steps:

1. Stop the geo-replication session if configured by executing the following command:

```
 # gluster volume geo-replication MASTER_VOL SLAVE_HOST::SLAVE_VOL
stop force
```

2. Stop the **glusterd** service on server0.example.com.

```
# service glusterd stop
```

Important

If **glusterd** crashes, there is no functionality impact to this crash as it occurs during the shutdown. For more information, see Section 24.3, “Resolving **glusterd** Crash”

3. Retrieve the UUID of the failed host (server0.example.com) from another peer in the Red Hat Gluster Storage Trusted Storage Pool by executing the following command:

```
# gluster peer status
Number of Peers: 1

Hostname: server0.example.com
Uuid: b5ab2ec3-5411-45fa-a30f-43bd04caf96b
State: Peer Rejected (Connected)
```

Note that the UUID of the failed host is **b5ab2ec3-5411-45fa-a30f-43bd04caf96b**

4. Edit the **glusterd.info** file in the new host (server0.example.com) and include the UUID of the host you retrieved in the previous step.

```
# cat /var/lib/glusterd/glusterd.info
UUID=b5ab2ec3-5411-45fa-a30f-43bd04caf96b
operating-version=30703
```

> **Note**
>
> The operating version of this node must be same as in other nodes of the trusted storage pool.

5. Create the peer file in the newly created host (server0.example.com) in /var/lib/glusterd/peers/<uuid-of-other-peer> with the name of the UUID of the other host (server1.example.com).

   UUID of the host can be obtained with the following:

   ```
   # gluster system:: uuid get
   ```

   Example 11.6. Example to obtain the UUID of a host

   ```
   For example,
   # gluster system:: uuid get
   UUID: 1d9677dc-6159-405e-9319-ad85ec030880
   ```

   In this case the UUID of other peer is **`1d9677dc-6159-405e-9319-ad85ec030880`**

6. Create a file **`/var/lib/glusterd/peers/1d9677dc-6159-405e-9319-ad85ec030880`** in server0.example.com, with the following command:

   ```
   # touch /var/lib/glusterd/peers/1d9677dc-6159-405e-9319-ad85ec030880
   ```

   The file you create must contain the following information:

   ```
   UUID=<uuid-of-other-node>
   state=3
   hostname=<hostname>
   ```

7. Continue to perform steps 12 to 18 as documented in the previous procedure.

11.11. Rebalancing Volumes

If a volume has been expanded or shrunk using the **`add-brick`** or **`remove-brick`** commands, the data on the volume needs to be rebalanced among the servers.

> **Note**
>
> In a non-replicated volume, all bricks should be online to perform the **`rebalance`** operation using the start option. In a replicated volume, at least one of the bricks in the replica should be online.

To rebalance a volume, use the following command on any of the servers:

```
# gluster volume rebalance VOLNAME start
```

For example:

```
# gluster volume rebalance test-volume start
Starting rebalancing on volume test-volume has been successful
```

When run without the **force** option, the rebalance command attempts to balance the space utilized across nodes. Files whose migration would cause the target node to have less available space than the source node are skipped. This results in linkto files being retained, which may cause slower access when a large number of linkto files are present.

Red Hat strongly recommends you to disconnect all the older clients before executing the rebalance command to avoid a potential data loss scenario.

Warning

The **Rebalance** command can be executed with the force option even when the older clients are connected to the cluster. However, this could lead to a data loss situation.

A **rebalance** operation with **force**, balances the data based on the layout, and hence optimizes or does away with the link files, but may lead to an imbalanced storage space used across bricks. This option is to be used only when there are a large number of link files in the system.

To rebalance a volume forcefully, use the following command on any of the servers:

```
# gluster volume rebalance VOLNAME start force
```

For example:

```
# gluster volume rebalance test-volume start force
Starting rebalancing on volume test-volume has been successful
```

11.11.1. Rebalance Throttling

The rebalance process uses multiple threads to ensure good performance during migration of multiple files. During multiple file migration, there can be a severe impact on storage system performance and a throttling mechanism is provided to manage it.

By default, the rebalance throttling is started in the **normal** mode. Configure the throttling modes to adjust the rate at which the files must be migrated

```
# gluster volume set VOLNAME rebal-throttle lazy|normal|aggressive
```

For example:

```
# gluster volume set test-volume rebal-throttle lazy
```

11.11.2. Displaying Rebalance Progress

To display the status of a volume rebalance operation, use the following command:

```
# gluster volume rebalance VOLNAME status
```

For example:

```
# gluster volume rebalance test-volume status
Node          Rebalanced size   scanned failures skipped status      run
time
              -files                                            in h:m:s
------------- ---------- ------ ------- -------- ------- ----------- ----
----
localhost   71962      70.3GB 380852  0        0       in progress 2:02:20
server1     70489      68.8GB 502185  0        0       in progress 2:02:20
server2     70704      69.0GB 507728  0        0       in progress 2:02:20
server3     71819      70.1GB 435611  0        0       in progress 2:02:20
Estimated time left for rebalance to complete :        2:50:24
```

A rebalance operation starts a rebalance process on each node of the volume. Each process is responsible for rebalancing the files on its own individual node. Each row of the rebalance status output describes the progress of the operation on a single node.

The following table describes the output of the rebalance status command:

Table 11.2. Rebalance Status Output Description

Property Name	Description
Node	The name of the node.
Rebalanced-files	The number of files that were successfully migrated.
size	The total size of the files that were migrated.
scanned	The number of files scanned on the node. This includes the files that were migrated.
failures	The number of files that could not be migrated because of errors.
skipped	The number of files which were skipped because of various errors or reasons.
status	The status of the rebalance operation on the node is **`in progress`**, **`completed`**, **`or failed`**.
run time in h:m:s	The amount of time for which the process has been running on the node.

The estimated time left for the rebalance to complete on all nodes is also displayed. The estimated time to complete is displayed only after the rebalance operation has been running for 10 minutes. In cases where the remaining time is extremely large, the estimated time to completion is displayed as **`>2 months`** and the user is advised to check again later.

The time taken to complete a rebalance operation depends on the number of files estimated to be on the bricks and the rate at which files are being processed by the rebalance process. This value is recalculated every time the rebalance status command is executed and becomes more accurate the longer rebalance has been running, and for large data sets. The calculation assumes that a file system partition contains a single brick.

A rebalance balance operation is considered complete when the status of every node is **`completed`**. For example:

```
  # gluster volume rebalance test-volume status
  Node       Rebalanced  size   scanned failures skipped status     run time
              -files                                           in h:m:s
  ---------- ---------- ----- ------- -------- ------- -----------  -----
```

```
---
  node2         0       0Bytes       0       0       0   completed
0:02:23
  node3       234      737.8KB    3350       0     257   completed
0:02:25
  node4         3        14.6K      71       0       6   completed
0:00:02
localhost   317        1.1MB   3484       0     155   completed   0:02:38
  volume rebalance: test-volume: success
```

With this release, details about the files that are skipped during rebalance operation can be obtained. Entries of all such files are available in the **`rebalance log`** with the message ID 109126. You can search for the message ID from the log file and get the list of all the skipped files:

For example:

```
# grep -i 109126 /var/log/glusterfs/test-volume-rebalance.log
[2018-03-15 09:14:30.203393] I [MSGID: 109126] [dht-
rebalance.c:2715:gf_defrag_migrate_single_file] 0-test-volume-dht: File
migration skipped for /linux-4.9.27/Documentation/ABI/stable/sysfs-fs-
orangefs.
[2018-03-15 09:14:31.262969] I [MSGID: 109126] [dht-
rebalance.c:2715:gf_defrag_migrate_single_file] 0-test-volume-dht: File
migration skipped for /linux-4.9.27/Documentation/ABI/stable/sysfs-devices.
[2018-03-15 09:14:31.842631] I [MSGID: 109126] [dht-
rebalance.c:2715:gf_defrag_migrate_single_file] 0-test-volume-dht: File
migration skipped for /linux-4.9.27/Documentation/ABI/stable/sysfs-devices-
system-cpu.
[2018-03-15 09:14:33.733728] I [MSGID: 109126] [dht-
rebalance.c:2715:gf_defrag_migrate_single_file] 0-test-volume-dht: File
migration skipped for /linux-4.9.27/Documentation/ABI/testing/sysfs-bus-fcoe.
[2018-03-15 09:14:35.576404] I [MSGID: 109126] [dht-
rebalance.c:2715:gf_defrag_migrate_single_file] 0-test-volume-dht: File
migration skipped for /linux-4.9.27/Documentation/ABI/testing/sysfs-bus-iio-
frequency-ad9523.
[2018-03-15 09:14:43.378480] I [MSGID: 109126] [dht-
rebalance.c:2715:gf_defrag_migrate_single_file] 0-test-volume-dht: File
migration skipped for /linux-4.9.27/Documentation/DocBook/kgdb.tmpl.
```

To know more about the failed files, search for 'migrate-data failed' in the rebalance.log file. However, the count for rebalance failed files will not match with "migrate-data failed" in the rebalance.log because the failed count includes all possible failures and just not file migration.

11.11.3. Stopping a Rebalance Operation

To stop a rebalance operation, use the following command:

```
# gluster volume rebalance VOLNAME stop
```

For example:

```
# gluster volume rebalance test-volume stop
Node          Rebalanced  size   scanned failures skipped status      run
time
              -files                                             in h:m:s
```

```
------------- ---------- ------- ------- -------- ------- ----------- ---
-----
localhost     106504     104.0GB 558111  0        0       stopped
3:02:24
server1       102299      99.9GB 725239  0        0       stopped
3:02:24
server2       102264      99.9GB 737364  0        0       stopped
3:02:24
server3       106813     104.3GB 646581  0        0       stopped
3:02:24
Estimated time left for rebalance to complete :        2:06:38
```

11.12. Setting up Shared Storage Volume

Features like Snapshot Scheduler, NFS Ganesha and geo-replication require a shared storage to be available across all nodes of the cluster. A gluster volume named **`gluster_shared_storage`** is made available for this purpose, and is facilitated by the following volume set option.

```
cluster.enable-shared-storage
```

This option accepts the following two values:

- **enable**

  When the volume set option is enabled, a gluster volume named **`gluster_shared_storage`** is created in the cluster, and is mounted at **`/var/run/gluster/shared_storage`** on all the nodes in the cluster.

  > **Note**
  >
  > - This option cannot be enabled if there is only one node present in the cluster, or if only one node is online in the cluster.
  > - The volume created is either a replica 2, or a replica 3 volume. This depends on the number of nodes which are online in the cluster at the time of enabling this option and each of these nodes will have one brick participating in the volume. The brick path participating in the volume is **`/var/lib/glusterd/ss_brick.`**
  > - The mount entry is also added to **`/etc/fstab`** as part of **`enable`**.
  > - Before enabling this feature make sure that there is no volume named **`gluster_shared_storage`** in the cluster. This volume name is reserved for internal use only

  After successfully setting up the shared storage volume, when a new node is added to the cluster, the shared storage is not mounted automatically on this node. Neither is the **`/etc/fstab`** entry added for the shared storage on this node. To make use of shared storage on this node, execute the following commands:

  ```
  # mount -t glusterfs <local node's ip>:gluster_shared_storage
  /var/run/gluster/shared_storage
  # cp /etc/fstab /var/run/gluster/fstab.tmp
  # echo "<local node's ip>:/gluster_shared_storage
  /var/run/gluster/shared_storage/ glusterfs defaults 0 0" >> /etc/fstab
  ```

- **disable**

When the volume set option is disabled, the **gluster_shared_storage** volume is unmounted on all the nodes in the cluster, and then the volume is deleted. The mount entry from **/etc/fstab** as part of **disable** is also removed.

For example:

```
# gluster volume set all cluster.enable-shared-storage enable
volume set: success
```

Important

After creating a cluster excute the following command on all nodes present in the cluster:

```
systemctl enable glusterfssharedstorage.service
```

This is applicable for Red Hat Enterprise Linux 7 only.

11.13. Stopping Volumes

To stop a volume, use the following command:

```
# gluster volume stop VOLNAME
```

For example, to stop test-volume:

```
# gluster volume stop test-volume
Stopping volume will make its data inaccessible. Do you want to continue?
(y/n) y
Stopping volume test-volume has been successful
```

11.14. Deleting Volumes

Important

Volumes must be unmounted and stopped before you can delete them. Ensure that you also remove entries relating to this volume from the **/etc/fstab** file after the volume has been deleted.

To delete a volume, use the following command:

```
# gluster volume delete VOLNAME
```

For example, to delete test-volume:

```
# gluster volume delete test-volume
Deleting volume will erase all information about the volume. Do you want to
continue? (y/n) y
Deleting volume test-volume has been successful
```

11.15. Managing Split-brain

Split-brain is a state of data inconsistency that occurs when different data sources in a cluster having different ideas about what the correct, current state of that data should be. This can happen because of servers in a network design, or a failure condition based on servers not communicating and synchronizing their data to each other.

In Red Hat Gluster Storage, split-brain is a term applicable to Red Hat Gluster Storage volumes in a replicate configuration. A file is said to be in split-brain when the copies of the same file in different bricks that constitute the replica-pair have mismatching data and/or metadata contents such that they are conflicting each other and automatic healing is not possible. In this scenario, you can decide which is the correct file (source) and which is the one that requires healing (sink) by inspecting at the mismatching files from the backend bricks.

The AFR translator in glusterFS makes use of extended attributes to keep track of the operations on a file. These attributes determine which brick is the correct source when a file requires healing. If the files are clean, the extended attributes are all zeroes indicating that no heal is necessary. When a heal is required, they are marked in such a way that there is a distinguishable source and sink and the heal can happen automatically. But, when a split-brain occurs, these extended attributes are marked in such a way that both bricks mark themselves as sources, making automatic healing impossible.

Split-brain occurs when a difference exists between multiple copies of the same file, and Red Hat Gluster Storage is unable to determine which version is correct. Applications are restricted from executing certain operations like *read* and *write* on the disputed file when split-brain happens. Attempting to access the files results in the application receiving an input/output error on the disputed file.

The three types of split-brain that occur in Red Hat Gluster Storage are:

- Data split-brain: Contents of the file under split-brain are different in different replica pairs and automatic healing is not possible.

  Red Hat allows the user to resolve Data split-brain from the mount point and from the CLI.

  For information on how to recover from data split-brain from the mount point, see Section 11.15.2.1, “ Recovering File Split-brain from the Mount Point”.

  For information on how to recover from data split-brain using CLIS, see Section 11.15.2.2, “Recovering File Split-brain from the gluster CLI”.

- Metadata split-brain: The metadata of the files like user defined extended attribute are different and automatic healing is not possible.

  Like Data split-brain, Metadata split-brain can also be resolved from both mount point and CLI.

  For information on how to recover from metadata split-brain from the mount point, see Section 11.15.2.1, “ Recovering File Split-brain from the Mount Point”.

  para>For information on how to recover from metadata split-brain using CLI, see Section 11.15.2.2, “Recovering File Split-brain from the gluster CLI”.

- Entry split-brain: Entry split-brain can be of two types:

- GlusterFS Internal File Identifier or GFID split-Brain: This happen when files or directories in different replica pairs have different GFIDs.

- Type Mismatch Split-Brain: This happen when the files/directories stored in replica pairs are of different types but with the same names.

Red Hat Gluster Storage 3.4 allows you to resolve GFID split-brain from gluster CLI. For more information, see Section 11.15.3, “Recovering GFID Split-brain from the gluster CLI”.

You can resolve split-brain manually by inspecting the file contents from the backend and deciding which is the true copy (source) and modifying the appropriate extended attributes such that healing can happen automatically.

11.15.1. Preventing Split-brain

To prevent split-brain in the trusted storage pool, you must configure server-side and client-side quorum.

11.15.1.1. Configuring Server-Side Quorum

The quorum configuration in a trusted storage pool determines the number of server failures that the trusted storage pool can sustain. If an additional failure occurs, the trusted storage pool will become unavailable. If too many server failures occur, or if there is a problem with communication between the trusted storage pool nodes, it is essential that the trusted storage pool be taken offline to prevent data loss.

After configuring the quorum ratio at the trusted storage pool level, you must enable the quorum on a particular volume by setting **`cluster.server-quorum-type`** volume option as **`server`**. For more information on this volume option, see Section 11.1, “Configuring Volume Options”.

Configuration of the quorum is necessary to prevent network partitions in the trusted storage pool. Network Partition is a scenario where, a small set of nodes might be able to communicate together across a functioning part of a network, but not be able to communicate with a different set of nodes in another part of the network. This can cause undesirable situations, such as split-brain in a distributed system. To prevent a split-brain situation, all the nodes in at least one of the partitions must stop running to avoid inconsistencies.

This quorum is on the server-side, that is, the **`glusterd`** service. Whenever the **`glusterd`** service on a machine observes that the quorum is not met, it brings down the bricks to prevent data split-brain. When the network connections are brought back up and the quorum is restored, the bricks in the volume are brought back up. When the quorum is not met for a volume, any commands that update the volume configuration or peer addition or detach are not allowed. It is to be noted that both, the **`glusterd`** service not running and the network connection between two machines being down are treated equally.

You can configure the quorum percentage ratio for a trusted storage pool. If the percentage ratio of the quorum is not met due to network outages, the bricks of the volume participating in the quorum in those nodes are taken offline. By default, the quorum is met if the percentage of active nodes is more than 50% of the total storage nodes. However, if the quorum ratio is manually configured, then the quorum is met only if the percentage of active storage nodes of the total storage nodes is greater than *or equal to* the set value.

To configure the quorum ratio, use the following command:

```
# gluster volume set all cluster.server-quorum-ratio PERCENTAGE
```

For example, to set the quorum to 51% of the trusted storage pool:

```
# gluster volume set all cluster.server-quorum-ratio 51%
```

In this example, the quorum ratio setting of 51% means that more than half of the nodes in the trusted storage pool must be online and have network connectivity between them at any given time. If a network disconnect happens to the storage pool, then the bricks running on those nodes are stopped to prevent further writes.

You must ensure to enable the quorum on a particular volume to participate in the server-side quorum by running the following command:

```
# gluster volume set VOLNAME cluster.server-quorum-type server
```

Important

For a two-node trusted storage pool, it is important to set the quorum ratio to be *greater than* 50% so that two nodes separated from each other do not both believe they have a quorum.

For a replicated volume with two nodes and one brick on each machine, if the server-side quorum is enabled and one of the nodes goes offline, the other node will also be taken offline because of the quorum configuration. As a result, the high availability provided by the replication is ineffective. To prevent this situation, a dummy node can be added to the trusted storage pool which does not contain any bricks. This ensures that even if one of the nodes which contains data goes offline, the other node will remain online. Note that if the dummy node and one of the data nodes goes offline, the brick on other node will be also be taken offline, and will result in data unavailability.

11.15.1.2. Configuring Client-Side Quorum

By default, when replication is configured, clients can modify files as long as at least one brick in the replica group is available. If network partitioning occurs, different clients are only able to connect to different bricks in a replica set, potentially allowing different clients to modify a single file simultaneously.

For example, imagine a three-way replicated volume is accessed by two clients, C1 and C2, who both want to modify the same file. If network partitioning occurs such that client C1 can only access brick B1, and client C2 can only access brick B2, then both clients are able to modify the file independently, creating split-brain conditions on the volume. The file becomes unusable, and manual intervention is required to correct the issue.

Client-side quorum allows administrators to set a minimum number of bricks that a client must be able to access in order to allow data in the volume to be modified. If client-side quorum is not met, files in the replica set are treated as read-only. This is useful when three-way replication is configured.

Client-side quorum is configured on a per-volume basis, and applies to all replica sets in a volume. If client-side quorum is not met for X of Y volume sets, only X volume sets are treated as read-only; the remaining volume sets continue to allow data modification.

Client-Side Quorum Options

cluster.quorum-count

The minimum number of bricks that must be available in order for writes to be allowed. This is set on a per-volume basis. Valid values are between `1` and the number of bricks in a replica set. This option is used by the `cluster.quorum-type` option to determine write behavior.

cluster.quorum-type

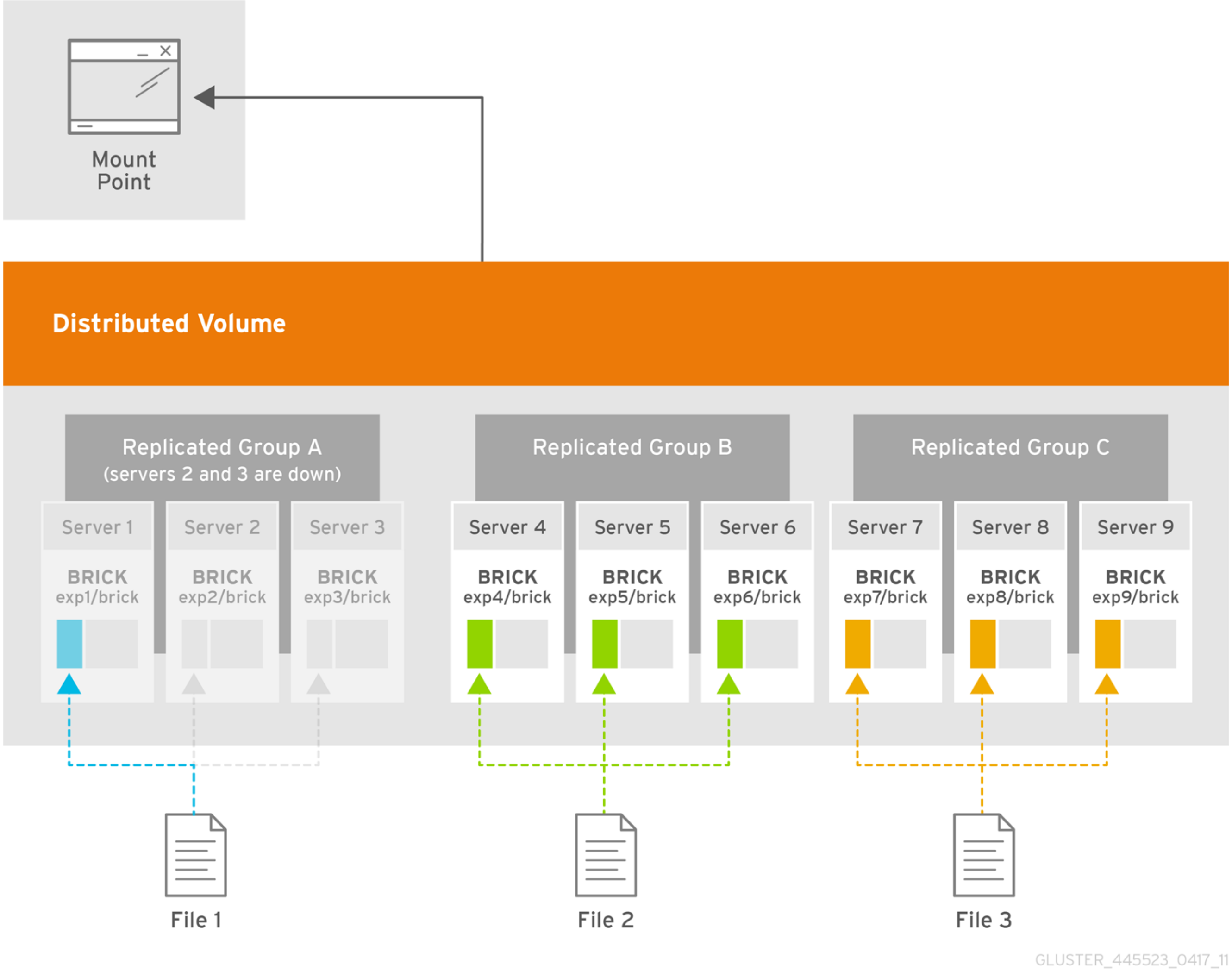
Determines when the client is allowed to write to a volume. Valid values are `fixed` and `auto`.

If `cluster.quorum-type` is `fixed`, writes are allowed as long as the number of bricks available in the replica set is greater than or equal to the value of the `cluster.quorum-count` option.

If **`cluster.quorum-type`** is **`auto`**, writes are allowed when at least 50% of the bricks in a replica set are be available. In a replica set with an even number of bricks, if exactly 50% of the bricks are available, the first brick in the replica set must be available in order for writes to continue.

In a three-way replication setup, it is recommended to set **`cluster.quorum-type`** to **`auto`** to avoid split-brains. If the quorum is not met, the replica pair becomes read-only.

Example 11.7. Client-Side Quorum



In the above scenario, when the client-side quorum is not met for replica group **A**, only replica group **A** becomes read-only. Replica groups **B** and **C** continue to allow data modifications.

Configure the client-side quorum using **`cluster.quorum-type`** and **`cluster.quorum-count`** options.

Important

When you integrate Red Hat Gluster Storage with Red Hat Enterprise Virtualization or Red Hat OpenStack, the client-side quorum is enabled when you run **`gluster volume set`** ***`VOLNAME`*** **`group virt`** command. If on a two replica set up, if the first brick in the replica pair is offline, virtual machines will be paused because quorum is not met and writes are disallowed.

Consistency is achieved at the cost of fault tolerance. If fault-tolerance is preferred over consistency, disable client-side quorum with the following command:

```
# gluster volume reset VOLNAME quorum-type
```

Example - Setting up server-side and client-side quorum to avoid split-brain scenario

This example provides information on how to set server-side and client-side quorum on a Distribute Replicate volume to avoid split-brain scenario. The configuration of this example has 2 X 2 (4 bricks) Distribute Replicate setup.

```
# gluster volume info testvol
Volume Name: testvol
Type: Distributed-Replicate
Volume ID: 0df52d58-bded-4e5d-ac37-4c82f7c89cfh
Status: Created
Number of Bricks: 2 x 2 = 4
Transport-type: tcp
Bricks:
Brick1: server1:/rhgs/brick1
Brick2: server2:/rhgs/brick2
Brick3: server3:/rhgs/brick3
Brick4: server4:/rhgs/brick4
```

Setting Server-side Quorum

Enable the quorum on a particular volume to participate in the server-side quorum by running the following command:

```
# gluster volume set VOLNAME cluster.server-quorum-type server
```

Set the quorum to 51% of the trusted storage pool:

```
# gluster volume set all cluster.server-quorum-ratio 51%
```

In this example, the quorum ratio setting of 51% means that more than half of the nodes in the trusted storage pool must be online and have network connectivity between them at any given time. If a network disconnect happens to the storage pool, then the bricks running on those nodes are stopped to prevent further writes.

Setting Client-side Quorum

Set the **`quorum-type`**option to **`auto`** to allow writes to the file only if the percentage of active replicate bricks is more than 50% of the total number of bricks that constitute that replica.

```
# gluster volume set VOLNAME quorum-type auto
```

In this example, as there are only two bricks in the replica pair, the first brick must be up and running to allow writes.

Important

Atleast n/2 bricks need to be up for the quorum to be met. If the number of bricks (**`n`**) in a replica set is an even number, it is mandatory that the **`n/2`** count must consist of the primary brick and it must be up and running. If **`n`** is an odd number, the **`n/2`** count can have any brick up and running, that is, the primary brick need not be up and running to allow writes.

11.15.2. Recovering from File Split-brain

You can recover from the data and meta-data split-brain using one of the following methods:

- See Section 11.15.2.1, “ Recovering File Split-brain from the Mount Point” for information on how to recover from data and meta-data split-brain from the mount point.

- See Section 11.15.2.2, “Recovering File Split-brain from the gluster CLI” for information on how to recover from data and meta-data split-brain using CLI

For information on resolving entry/type-mismatch split-brain, see Chapter 26, *Manually Recovering File Split-brain* .

11.15.2.1. Recovering File Split-brain from the Mount Point

Steps to recover from a split-brain from the mount point

1. You can use a set of **`getfattr`** and **`setfattr`** commands to detect the data and meta-data split-brain status of a file and resolve split-brain from the mount point.

   Important

   This process for split-brain resolution from mount will not work on NFS mounts as it does not provide extended attributes support.

   In this example, the **`test-volume`** volume has bricks **`brick0`**, **`brick1`**, **`brick2`** and **`brick3`**.

   ```
   # gluster volume info test-volume
   Volume Name: test-volume
   Type: Distributed-Replicate
   Status: Started
   Number of Bricks: 2 x 2 = 4
   Transport-type: tcp
   Bricks:
   Brick1: test-host:/rhgs/brick0
   Brick2: test-host:/rhgs/brick1
   Brick3: test-host:/rhgs/brick2
   Brick4: test-host:/rhgs/brick3
   ```

   Directory structure of the bricks is as follows:

```
# tree -R /test/b?
/rhgs/brick0
├── dir
│   └── a
└── file100

/rhgs/brick1
├── dir
│   └── a
└── file100

/rhgs/brick2
├── dir
├── file1
├── file2
└── file99

/rhgs/brick3
├── dir
├── file1
├── file2
└── file99
```

In the following output, some of the files in the volume are in split-brain.

```
# gluster volume heal test-volume info split-brain
Brick test-host:/rhgs/brick0/
/file100
/dir
Number of entries in split-brain: 2

Brick test-host:/rhgs/brick1/
/file100
/dir
Number of entries in split-brain: 2

Brick test-host:/rhgs/brick2/
/file99
<gfid:5399a8d1-aee9-4653-bb7f-606df02b3696>
Number of entries in split-brain: 2

Brick test-host:/rhgs/brick3/
<gfid:05c4b283-af58-48ed-999e-4d706c7b97d5>
<gfid:5399a8d1-aee9-4653-bb7f-606df02b3696>
Number of entries in split-brain: 2
```

To know data or meta-data split-brain status of a file:

```
# getfattr -n replica.split-brain-status <path-to-file>
```

The above command executed from mount provides information if a file is in data or meta-data split-brain. This command is not applicable to entry/type-mismatch split-brain.

For example,

- **`file100`** is in meta-data split-brain. Executing the above mentioned command for **`file100`** gives :

  ```
  # getfattr -n replica.split-brain-status file100
  # file: file100
  replica.split-brain-status="data-split-brain:no    metadata-split-
  brain:yes    Choices:test-client-0,test-client-1"
  ```

- **`file1`** is in data split-brain.

  ```
  # getfattr -n replica.split-brain-status file1
  # file: file1
  replica.split-brain-status="data-split-brain:yes    metadata-split-
  brain:no    Choices:test-client-2,test-client-3"
  ```

- **`file99`** is in both data and meta-data split-brain.

  ```
  # getfattr -n replica.split-brain-status file99
  # file: file99
  replica.split-brain-status="data-split-brain:yes    metadata-split-
  brain:yes    Choices:test-client-2,test-client-3"
  ```

- **`dir`** is in **`entry/type-mismatch`** split-brain but as mentioned earlier, the above command is does not display if the file is in **`entry/type-mismatch`** split-brain. Hence, the command displays **`The file is not under data or metadata split-brain`**. For information on resolving entry/type-mismatch split-brain, see Chapter 26, *Manually Recovering File Split-brain* .

  ```
  # getfattr -n replica.split-brain-status dir
  # file: dir
  replica.split-brain-status="The file is not under data or metadata
  split-brain"
  ```

- **`file2`** is not in any kind of split-brain.

  ```
  # getfattr -n replica.split-brain-status file2
  # file: file2
  replica.split-brain-status="The file is not under data or metadata
  split-brain"
  ```

2. **Analyze the files in data and meta-data split-brain and resolve the issue**

   When you perform operations like **`cat`**, **`getfattr`**, and more from the mount on files in split-brain, it throws an input/output error. For further analyzing such files, you can use **`setfattr`** command.

   ```
   # setfattr -n replica.split-brain-choice -v "choiceX" <path-to-file>
   ```

   Using this command, a particular brick can be chosen to access the file in split-brain.

   For example,

   `file1` is in data-split-brain and when you try to read from the file, it throws input/output error.

   ```
   # cat file1
   cat: file1: Input/output error
   ```

Split-brain choices provided for file1 were **`test-client-2`** and **`test-client-3`**.

Setting **`test-client-2`** as split-brain choice for file1 serves reads from **`b2`** for the file.

```
# setfattr -n replica.split-brain-choice -v test-client-2 file1
```

Now, you can perform operations on the file. For example, read operations on the file:

```
# cat file1
xyz
```

Similarly, to inspect the file from other choice, **`replica.split-brain-choice`** is to be set to **`test-client-3`**.

Trying to inspect the file from a wrong choice errors out. You can undo the split-brain-choice that has been set, the above mentioned **`setfattr`** command can be used with **none** as the value for extended attribute.

For example,

```
# setfattr -n replica.split-brain-choice -v none file1
```

Now performing **`cat`** operation on the file will again result in input/output error, as before.

```
# cat file
cat: file1: Input/output error
```

After you decide which brick to use as a source for resolving the split-brain, it must be set for the healing to be done.

```
# setfattr -n replica.split-brain-heal-finalize -v <heal-choice>
<path-to-file>
```

Example

```
# setfattr -n replica.split-brain-heal-finalize -v test-client-2 file1
```

The above process can be used to resolve data and/or meta-data split-brain on all the files.

Setting the split-brain-choice on the file

After setting the split-brain-choice on the file, the file can be analyzed only for five minutes. If the duration of analyzing the file needs to be increased, use the following command and set the required time in **`timeout-in-minute`** argument.

```
# setfattr -n replica.split-brain-choice-timeout -v <timeout-in-
minutes> <mount_point/file>
```

This is a global timeout and is applicable to all files as long as the mount exists. The timeout need not be set each time a file needs to be inspected but for a new mount it will have to be set again for the first time. This option becomes invalid if the operations like add-brick or remove-brick are performed.

Note

If **`fopen-keep-cache`** FUSE mount option is disabled, then inode must be invalidated each time before selecting a new **`replica.split-brain-choice`** to inspect a file using the following command:

```
# setfattr -n inode-invalidate -v 0 <path-to-file>
```

11.15.2.2. Recovering File Split-brain from the gluster CLI

You can resolve the split-brain from the gluster CLI by the following ways:

- Use bigger-file as source
- Use the file with latest mtime as source
- Use one replica as source for a particular file
- Use one replica as source for all files

Note

The **`entry/type-mismatch`** split-brain resolution is not supported using CLI. For information on resolving **`entry/type-mismatch`** split-brain, see Chapter 26, *Manually Recovering File Split-brain* .

Selecting the bigger-file as source

This method is useful for per file healing and where you can decided that the file with bigger size is to be considered as source.

1. Run the following command to obtain the list of files that are in split-brain:

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

```
Brick <hostname:brickpath-b1>
<gfid:aaca219f-0e25-4576-8689-3bfd93ca70c2>
<gfid:39f301ae-4038-48c2-a889-7dac143e82dd>
<gfid:c3c94de2-232d-4083-b534-5da17fc476ac>
Number of entries in split-brain: 3

Brick <hostname:brickpath-b2>
/dir/file1
/dir
/file4
Number of entries in split-brain: 3
```

From the command output, identify the files that are in split-brain.

You can find the differences in the file size and md5 checksums by performing a stat and md5 checksums on the file from the bricks. The following is the stat and md5 checksum output of a file:

```
On brick b1:
# stat b1/dir/file1
  File: ‘b1/dir/file1’
  Size: 17              Blocks: 16         IO Block: 4096   regular
file
Device: fd03h/64771d    Inode: 919362      Links: 2
Access: (0644/-rw-r--r--)  Uid: (    0/    root)   Gid: (    0/
root)
Access: 2015-03-06 13:55:40.149897333 +0530
Modify: 2015-03-06 13:55:37.206880347 +0530
Change: 2015-03-06 13:55:37.206880347 +0530
 Birth: -

# md5sum b1/dir/file1
040751929ceabf77c3c0b3b662f341a8  b1/dir/file1

On brick b2:
# stat b2/dir/file1
  File: ‘b2/dir/file1’
  Size: 13              Blocks: 16         IO Block: 4096   regular
file
Device: fd03h/64771d    Inode: 919365      Links: 2
Access: (0644/-rw-r--r--)  Uid: (    0/    root)   Gid: (    0/
root)
Access: 2015-03-06 13:54:22.974451898 +0530
Modify: 2015-03-06 13:52:22.910758923 +0530
Change: 2015-03-06 13:52:22.910758923 +0530
 Birth: -

# md5sum b2/dir/file1
cb11635a45d45668a403145059c2a0d5  b2/dir/file1
```

You can notice the differences in the file size and md5 checksums.

2. Execute the following command along with the full file name as seen from the root of the volume (or) the gfid-string representation of the file, which is displayed in the heal info command's output.

   ```
   # gluster volume heal <VOLNAME> split-brain bigger-file <FILE>
   ```

   For example,

   ```
   # gluster volume heal test-volume split-brain bigger-file /dir/file1
   Healed /dir/file1.
   ```

After the healing is complete, the md5sum and file size on both bricks must be same. The following is a sample output of the stat and md5 checksums command after completion of healing the file.

```
On brick b1:
# stat b1/dir/file1
  File: ‘b1/dir/file1’
  Size: 17              Blocks: 16         IO Block: 4096   regular file
Device: fd03h/64771d    Inode: 919362      Links: 2
```

```
Access: (0644/-rw-r--r--)  Uid: (    0/    root)   Gid: (    0/    root)
Access: 2015-03-06 14:17:27.752429505 +0530
Modify: 2015-03-06 13:55:37.206880347 +0530
Change: 2015-03-06 14:17:12.880343950 +0530
 Birth: -

# md5sum b1/dir/file1
040751929ceabf77c3c0b3b662f341a8  b1/dir/file1

On brick b2:
# stat b2/dir/file1
  File: ‘b2/dir/file1’
  Size: 17              Blocks: 16         IO Block: 4096   regular file
Device: fd03h/64771d    Inode: 919365      Links: 2
Access: (0644/-rw-r--r--)  Uid: (    0/    root)   Gid: (    0/    root)
Access: 2015-03-06 14:17:23.249403600 +0530
Modify: 2015-03-06 13:55:37.206880000 +0530
Change: 2015-03-06 14:17:12.881343955 +0530
 Birth: -

# md5sum b2/dir/file1
040751929ceabf77c3c0b3b662f341a8  b2/dir/file1
```

Selecting the file with latest mtime as source

This method is useful for per file healing and if you want the file with latest mtime has to be considered as source.

1. Run the following command to obtain the list of files that are in split-brain:

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

   ```
   Brick <hostname:brickpath-b1>
   <gfid:aaca219f-0e25-4576-8689-3bfd93ca70c2>
   <gfid:39f301ae-4038-48c2-a889-7dac143e82dd>
   <gfid:c3c94de2-232d-4083-b534-5da17fc476ac>
   Number of entries in split-brain: 3

   Brick <hostname:brickpath-b2>
   /dir/file1
   /dir
   /file4
   Number of entries in split-brain: 3
   ```

   From the command output, identify the files that are in split-brain.

   You can find the differences in the file size and md5 checksums by performing a stat and md5 checksums on the file from the bricks. The following is the stat and md5 checksum output of a file:

   ```
   On brick b1:

    stat b1/file4
     File: ‘b1/file4’
       Size: 4               Blocks: 16         IO Block: 4096   regular
   file
   ```

```
Device: fd03h/64771d    Inode: 919356      Links: 2
Access: (0644/-rw-r--r--)  Uid: (    0/    root)   Gid: (    0/
root)
Access: 2015-03-06 13:53:19.417085062 +0530
Modify: 2015-03-06 13:53:19.426085114 +0530
Change: 2015-03-06 13:53:19.426085114 +0530
 Birth: -


# md5sum b1/file4
b6273b589df2dfdbd8fe35b1011e3183  b1/file4

On brick b2:

# stat b2/file4
  File: ‘b2/file4’
  Size: 4               Blocks: 16         IO Block: 4096   regular
file
Device: fd03h/64771d    Inode: 919358      Links: 2
Access: (0644/-rw-r--r--)  Uid: (    0/    root)   Gid: (    0/
root)
Access: 2015-03-06 13:52:35.761833096 +0530
Modify: 2015-03-06 13:52:35.769833142 +0530
Change: 2015-03-06 13:52:35.769833142 +0530
 Birth: -


# md5sum b2/file4
0bee89b07a248e27c83fc3d5951213c1  b2/file4
```

You can notice the differences in the md5 checksums, and the modify time.

2. Execute the following command

```
# gluster volume heal <VOLNAME> split-brain latest-mtime <FILE>
```

In this command, *FILE* can be either the full file name as seen from the root of the volume or the gfid-string representation of the file.

For example,

```
#gluster volume heal test-volume split-brain latest-mtime /file4
Healed /file4
```

After the healing is complete, the md5 checksum, file size, and modify time on both bricks must be same. The following is a sample output of the stat and md5 checksums command after completion of healing the file. You can notice that the file has been healed using the brick having the latest mtime (brick b1, in this example) as the source.

```
On brick b1:
# stat b1/file4
  File: ‘b1/file4’
  Size: 4               Blocks: 16         IO Block: 4096   regular
file
Device: fd03h/64771d    Inode: 919356      Links: 2
```

```
Access: (0644/-rw-r--r--)  Uid: (    0/    root)   Gid: (    0/
root)
Access: 2015-03-06 14:23:38.944609863 +0530
Modify: 2015-03-06 13:53:19.426085114 +0530
Change: 2015-03-06 14:27:15.058927962 +0530
 Birth: -

# md5sum b1/file4
b6273b589df2dfdbd8fe35b1011e3183  b1/file4

On brick b2:
# stat b2/file4
 File: ‘b2/file4’
   Size: 4               Blocks: 16         IO Block: 4096   regular
file
Device: fd03h/64771d    Inode: 919358      Links: 2
Access: (0644/-rw-r--r--)  Uid: (    0/    root)   Gid: (    0/
root)
Access: 2015-03-06 14:23:38.944609000 +0530
Modify: 2015-03-06 13:53:19.426085000 +0530
Change: 2015-03-06 14:27:15.059927968 +0530
 Birth:

# md5sum b2/file4
b6273b589df2dfdbd8fe35b1011e3183  b2/file4
```

Selecting one replica as source for a particular file

This method is useful if you know which file is to be considered as source.

1. Run the following command to obtain the list of files that are in split-brain:

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

```
Brick <hostname:brickpath-b1>
<gfid:aaca219f-0e25-4576-8689-3bfd93ca70c2>
<gfid:39f301ae-4038-48c2-a889-7dac143e82dd>
<gfid:c3c94de2-232d-4083-b534-5da17fc476ac>
Number of entries in split-brain: 3

Brick <hostname:brickpath-b2>
/dir/file1
/dir
/file4
Number of entries in split-brain: 3
```

From the command output, identify the files that are in split-brain.

You can find the differences in the file size and md5 checksums by performing a stat and md5 checksums on the file from the bricks. The following is the stat and md5 checksum output of a file:

```
On brick b1:

 stat b1/file4
  File: ‘b1/file4’
  Size: 4               Blocks: 16         IO Block: 4096   regular
file
Device: fd03h/64771d    Inode: 919356      Links: 2
Access: (0644/-rw-r--r--)  Uid: (    0/    root)   Gid: (    0/
root)
Access: 2015-03-06 13:53:19.417085062 +0530
Modify: 2015-03-06 13:53:19.426085114 +0530
Change: 2015-03-06 13:53:19.426085114 +0530
 Birth: -

# md5sum b1/file4
b6273b589df2dfdbd8fe35b1011e3183  b1/file4

On brick b2:

# stat b2/file4
  File: ‘b2/file4’
  Size: 4               Blocks: 16         IO Block: 4096   regular
file
Device: fd03h/64771d    Inode: 919358      Links: 2
Access: (0644/-rw-r--r--)  Uid: (    0/    root)   Gid: (    0/
root)
Access: 2015-03-06 13:52:35.761833096 +0530
Modify: 2015-03-06 13:52:35.769833142 +0530
Change: 2015-03-06 13:52:35.769833142 +0530
 Birth: -

# md5sum b2/file4
0bee89b07a248e27c83fc3d5951213c1  b2/file4
```

You can notice the differences in the file size and md5 checksums.

2. Execute the following command

```
# gluster volume heal <VOLNAME> split-brain source-brick
<HOSTNAME:BRICKNAME> <FILE>
```

In this command, *FILE* present in *<HOSTNAME:BRICKNAME>* is taken as source for healing.

For example,

```
# gluster volume heal test-volume split-brain source-brick test-
host:b1 /file4
Healed /file4
```

After the healing is complete, the md5 checksum and file size on both bricks must be same. The following is a sample output of the stat and md5 checksums command after completion of healing the file.

```
On brick b1:
# stat b1/file4
  File: ‘b1/file4’
  Size: 4               Blocks: 16         IO Block: 4096   regular
file
Device: fd03h/64771d    Inode: 919356      Links: 2
```

```
Access: (0644/-rw-r--r--)  Uid: (    0/    root)   Gid: (    0/
root)
Access: 2015-03-06 14:23:38.944609863 +0530
Modify: 2015-03-06 13:53:19.426085114 +0530
Change: 2015-03-06 14:27:15.058927962 +0530
 Birth: -

# md5sum b1/file4
b6273b589df2dfdbd8fe35b1011e3183  b1/file4

On brick b2:
# stat b2/file4
 File: ‘b2/file4’
  Size: 4               Blocks: 16         IO Block: 4096   regular
file
Device: fd03h/64771d    Inode: 919358      Links: 2
Access: (0644/-rw-r--r--)  Uid: (    0/    root)   Gid: (    0/
root)
Access: 2015-03-06 14:23:38.944609000 +0530
Modify: 2015-03-06 13:53:19.426085000 +0530
Change: 2015-03-06 14:27:15.059927968 +0530
 Birth: -

# md5sum b2/file4
b6273b589df2dfdbd8fe35b1011e3183  b2/file4
```

Selecting one replica as source for all files

This method is useful if you know want to use a particular brick as a source for the split-brain files in that replica pair.

1. Run the following command to obtain the list of files that are in split-brain:

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

   From the command output, identify the files that are in split-brain.

2. Execute the following command

   ```
   # gluster volume heal <VOLNAME> split-brain source-brick
   <HOSTNAME:BRICKNAME>
   ```

   In this command, for all the files that are in split-brain in this replica, *<HOSTNAME:BRICKNAME>* is taken as source for healing.

   For example,

   ```
   # gluster volume heal test-volume split-brain source-brick test-
   host:b1
   ```

11.15.3. Recovering GFID Split-brain from the gluster CLI

With this release, Red Hat Gluster Storage allows you to resolve GFID split-brain from the gluster CLI.

You can use one of the following policies to resolve GFID split-brain:

- Use bigger-file as source
- Use the file with latest mtime as source
- Use one replica as source for a particular file

Note

The entry/type-mismatch split-brain resolution is not supported using CLI. For information on resolving entry/type-mismatch split-brain, see Chapter 26, *Manually Recovering File Split-brain* .

Selecting the bigger-file as source

This method is useful for per file healing and where you can decided that the file with bigger size is to be considered as source.

1. Run the following command to obtain the path of the file that is in split-brain:

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

   From the output, identify the files for which file operations performed from the client failed with input/output error.

   For example,

   ```
   # gluster volume heal 12 info split-brain
   ```

   ```
   Brick 10.70.47.45:/bricks/brick2/b0
   /f5
   / - Is in split-brain

   Status: Connected
   Number of entries: 2

   Brick 10.70.47.144:/bricks/brick2/b1
   /f5
   / - Is in split-brain

   Status: Connected
   Number of entries: 2
   ```

   In the above command, *12* is the volume name, *b0* and *b1* are the bricks.

2. Execute the below command on the brick to fetch information if a file is in GFID split-brain. The **getfattr** command is used to obtain and verify the AFR changelog extended attributes of the files.

   ```
       #getfattr -d -e hex -m. <path-to-file>
   ```

   For example,

   ```
   On brick /b0

   # getfattr -d -m . -e hex /bricks/brick2/b0/f5
   ```

```
getfattr: Removing leading '/' from absolute path names
# file: bricks/brick2/b0/f5
security.selinux=0x73797374656d5f753a6f626a6563745f723a676c75737465726
45f627269636b5f743a733000
trusted.afr.12-client-1=0x000000020000000100000000
trusted.afr.dirty=0x000000000000000000000000
trusted.gfid=0xce0a9956928e40afb78e95f78defd64f
trusted.gfid2path.9cde09916eabc845=0x30303030303030302d303030302d30303
0302d303030302d303030303030303030303030312f6635



On brick /b1

# getfattr -d -m . -e hex /bricks/brick2/b1/f5
getfattr: Removing leading '/' from absolute path names
# file: bricks/brick2/b1/f5
security.selinux=0x73797374656d5f753a6f626a6563745f723a676c75737465726
45f627269636b5f743a733000
trusted.afr.12-client-0=0x000000020000000100000000
trusted.afr.dirty=0x000000000000000000000000
trusted.gfid=0x9563544118653550e888ab38c232e0c
trusted.gfid2path.9cde09916eabc845=0x30303030303030302d303030302d30303
0302d303030302d303030303030303030303030312f6635
```

You can notice the difference in GFID for the file *f5* in both the bricks.

You can find the differences in the file size by executing **`stat`** command on the file from the bricks. The following is the output of the file *f5* in bricks *b0* and *b1*:

```
On brick /b0

# stat /bricks/brick2/b0/f5
File: ‘/bricks/brick2/b0/f5’
Size: 15            Blocks: 8          IO Block: 4096   regular file
Device: fd15h/64789d    Inode: 67113350    Links: 2
Access: (0644/-rw-r--r--)  Uid: (    0/    root)   Gid: (    0/
root)
Context: system_u:object_r:glusterd_brick_t:s0
Access: 2018-08-29 20:46:26.353751073 +0530
Modify: 2018-08-29 20:46:26.361751203 +0530
Change: 2018-08-29 20:47:16.363751236 +0530
Birth: -



On brick /b1

# stat /bricks/brick2/b1/f5
File: ‘/bricks/brick2/b1/f5’
Size: 2             Blocks: 8          IO Block: 4096   regular file
Device: fd15h/64789d    Inode: 67111750    Links: 2
Access: (0644/-rw-r--r--)  Uid: (    0/    root)   Gid: (    0/
root)
Context: system_u:object_r:glusterd_brick_t:s0
```

```
Access: 2018-08-29 20:44:56.153301616 +0530
Modify: 2018-08-29 20:44:56.161301745 +0530
Change: 2018-08-29 20:44:56.162301761 +0530
Birth: -
```

3. Execute the following command along with the full filename as seen from the root of the volume which is displayed in the **heal info** command's output:

   ```
   #gluster volume heal VOLNAME split-brain bigger-file FILE
   ```

   For example,

   ```
   # gluster volume heal12 split-brain bigger-file /f5
   GFID split-brain resolved for file /f5
   ```

   After the healing is complete, the file size on both bricks must be the same as that of the file which had the bigger size. The following is a sample output of the **getfattr**command after completion of healing the file.

   ```
   On brick /b0

   # getfattr -d -m . -e hex /bricks/brick2/b0/f5
   getfattr: Removing leading '/' from absolute path names
   # file: bricks/brick2/b0/f5
   security.selinux=0x73797374656d5f753a6f626a6563745f723a676c75737465726
   45f627269636b5f743a733000
   trusted.gfid=0xce0a9956928e40afb78e95f78defd64f
   trusted.gfid2path.9cde09916eabc845=0x30303030303030302d303030302d30303
   0302d303030302d3030303030303030303030312f6635



   On brick /b1

   # getfattr -d -m . -e hex /bricks/brick2/b1/f5
   getfattr: Removing leading '/' from absolute path names
   # file: bricks/brick2/b1/f5
   security.selinux=0x73797374656d5f753a6f626a6563745f723a676c75737465726
   45f627269636b5f743a733000
   trusted.gfid=0xce0a9956928e40afb78e95f78defd64f
   trusted.gfid2path.9cde09916eabc845=0x30303030303030302d303030302d30303
   0302d303030302d3030303030303030303030312f6635
   ```

Selecting the file with latest mtime as source

This method is useful for per file healing and if you want the file with latest mtime has to be considered as source.

1. Run the following command to obtain the list of files that are in split-brain:

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

   From the output, identify the files for which file operations performed from the client failed with input/output error.

For example,

```
# gluster volume heal 12 info split-brain
```

```
Brick 10.70.47.45:/bricks/brick2/b0
/f4
/ - Is in split-brain

Status: Connected
Number of entries: 2

Brick 10.70.47.144:/bricks/brick2/b1
/f4
/ - Is in split-brain

Status: Connected
Number of entries: 2
```

In the above command, *12* is the volume name, *b0* and *b1* are the bricks.

2. The below command executed from backend provides information if a file is in GFID split-brain.

```
# getfattr -d -e hex -m. <path-to-file>
```

For example,

```
On brick /b0

# getfattr -d -m . -e hex /bricks/brick2/b0/f4
getfattr: Removing leading '/' from absolute path names
# file: bricks/brick2/b0/f4
security.selinux=0x73797374656d5f753a6f626a6563745f723a676c75737465726
45f627269636b5f743a733000
trusted.afr.12-client-1=0x000000020000000100000000
trusted.afr.dirty=0x000000000000000000000000
trusted.gfid=0xb66b66d07b315f3c9cffac2fb6422a28
trusted.gfid2path.364f55367c7bd6f4=0x30303030303030302d303030302d30303
0302d303030302d30303030303030303030312f6634



On brick /b1

# getfattr -d -m . -e hex /bricks/brick2/b1/f4
getfattr: Removing leading '/' from absolute path names
# file: bricks/brick2/b1/f4
security.selinux=0x73797374656d5f753a6f626a6563745f723a676c75737465726
45f627269636b5f743a733000
trusted.afr.12-client-0=0x000000020000000100000000
trusted.afr.dirty=0x000000000000000000000000
trusted.gfid=0x87242f808c6e56a007ef7d49d197acff
trusted.gfid2path.364f55367c7bd6f4=0x30303030303030302d303030302d30303
0302d303030302d30303030303030303030312f6634
```

You can notice the difference in GFID for the file *f4* in both the bricks.

You can find the difference in the modify time by executing **stat**command on the file from the bricks. The following is the output of the file *f4* in bricks *b0* and *b1*:

```
On brick /b0

# stat /bricks/brick2/b0/f4
File: ‘/bricks/brick2/b0/f4’
Size: 14            Blocks: 8          IO Block: 4096   regular file
Device: fd15h/64789d    Inode: 67113349    Links: 2
Access: (0644/-rw-r--r--)  Uid: (    0/    root)   Gid: (    0/
root)
Context: system_u:object_r:glusterd_brick_t:s0
Access: 2018-08-29 20:57:38.913629991 +0530
Modify: 2018-08-29 20:57:38.921630122 +0530
Change: 2018-08-29 20:57:38.923630154 +0530
Birth: -



On brick /b1

# stat /bricks/brick2/b1/f4
File: ‘/bricks/brick2/b1/f4’
Size: 2             Blocks: 8          IO Block: 4096   regular file
Device: fd15h/64789d    Inode: 67111749    Links: 2
Access: (0644/-rw-r--r--)  Uid: (    0/    root)   Gid: (    0/
root)
Context: system_u:object_r:glusterd_brick_t:s0
Access: 2018-08-24 20:54:50.953217256 +0530
Modify: 2018-08-24 20:54:50.961217385 +0530
Change: 2018-08-24 20:54:50.962217402 +0530
Birth: -
```

3. Execute the following command:

```
#gluster volume healVOLNAME split-brain latest-mtime FILE
```

For example,

```
# gluster volume heal 12 split-brain latest-mtime /f4
GFID split-brain resolved for file /f4
```

After the healing is complete, the GFID of the files on both bricks must be same. The following is a sample output of the **getfattr** command after completion of healing the file. You can notice that the file has been healed using the brick having the latest mtime as the source.

```
On brick /b0

# getfattr -d -m . -e hex /bricks/brick2/b0/f4
getfattr: Removing leading '/' from absolute path names
# file: bricks/brick2/b0/f4
security.selinux=0x73797374656d5f753a6f626a6563745f723a676c75737465726
45f627269636b5f743a733000
trusted.gfid=0xb66b66d07b315f3c9cffac2fb6422a28
trusted.gfid2path.364f55367c7bd6f4=0x30303030303030302d303030302d30303
```

```
0302d303030302d3030303030303030303030312f6634



On brick /b1

# getfattr -d -m . -e hex /bricks/brick2/b1/f4
getfattr: Removing leading '/' from absolute path names
# file: bricks/brick2/b1/f4
security.selinux=0x73797374656d5f753a6f626a6563745f723a676c75737465726
45f627269636b5f743a733000
trusted.gfid=0xb66b66d07b315f3c9cffac2fb6422a28
trusted.gfid2path.364f55367c7bd6f4=0x30303030303030302d303030302d30303
0302d303030302d3030303030303030303030312f6634
```

Selecting one replica as source for a particular file

This method is useful if you know which file is to be considered as source.

1. Run the following command to obtain the list of files that are in split-brain:

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

   From the output, identify the files for which file operations performed from the client failed with input/output error.

   For example,

   ```
   # gluster volume heal 12 info split-brain
   ```

   ```
   Brick 10.70.47.45:/bricks/brick2/b0
   /f3
   / - Is in split-brain

   Status: Connected
   Number of entries: 2

   Brick 10.70.47.144:/bricks/brick2/b1
   /f3
   / - Is in split-brain

   Status: Connected
   Number of entries: 2
   ```

   In the above command, *12* is the volume name, *b0* and *b1* are the bricks.

Note

With one replica as source option, there is no way to resolve all the GFID split-brain in one shot by not specifying any file-path in the CLI as done for data/metadata split-brain resolutions.

For each file in GFID split-brain, you have to run the **heal** command separately.

2. The below command executed from backend provides information if a file is in GFID split-brain.

   ```
   # getfattr -d -e hex -m. <path-to-file>
   ```

   For example,

   ```
   # getfattr -d -m . -e hex /bricks/brick2/b0/f3
   On brick /b0

   getfattr: Removing leading '/' from absolute path names
   # file: bricks/brick2/b0/f3
   security.selinux=0x73797374656d5f753a6f626a6563745f723a676c75737465726
   45f627269636b5f743a733000
   trusted.afr.12-client-1=0x000000020000000100000000
   trusted.afr.dirty=0x000000000000000000000000
   trusted.gfid=0x9d542fb1b3b15837a2f7f9dcdf5d6ee8
   trusted.gfid2path.364f55367c7bd6f4=0x30303030303030302d303030302d30303
   0302d303030302d30303030303030303030312f6634


   On brick /b1

   # getfattr -d -m . -e hex /bricks/brick2/b1/f3
   getfattr: Removing leading '/' from absolute path names
   # file: bricks/brick2/b0/f3
   security.selinux=0x73797374656d5f753a6f626a6563745f723a676c75737465726
   45f627269636b5f743a733000
   trusted.afr.12-client-1=0x000000020000000100000000
   trusted.afr.dirty=0x000000000000000000000000
   trusted.gfid=0xc90d9b0f65f6530b95b9f3f8334033df
   trusted.gfid2path.364f55367c7bd6f4=0x30303030303030302d303030302d30303
   0302d303030302d30303030303030303030312f6634
   ```

   You can notice the difference in GFID for the file *f3* in both the bricks.

3. Execute the following command:

   ```
   #gluster volume heal VOLNAME split-brain source-brick HOSTNAME :
   export-directory-absolute-path FILE
   ```

   In this command, FILE present in *HOSTNAME* : *export-directory-absolute-path* is taken as source for healing.

   For example,

```
# gluster volume heal 12 split-brain source-brick
10.70.47.144:/bricks/brick2/b1 /f3
GFID split-brain resolved for file /f3
```

After the healing is complete, the GFID of the file on both the bricks should be same as that of the file which had bigger size. The following is a sample output of the **`getfattr`** command after the file is healed.

```
On brick /b0

# getfattr -d -m . -e hex /bricks/brick2/b0/f3
getfattr: Removing leading '/' from absolute path names
# file: bricks/brick2/b0/f3
security.selinux=0x73797374656d5f753a6f626a6563745f723a676c75737465726
45f627269636b5f743a733000
trusted.gfid=0x90d9b0f65f6530b95b9f3f8334033df
trusted.gfid2path.364f55367c7bd6f4=0x30303030303030302d303030302d30303
0302d303030302d30303030303030303030312f6634



On brick /b1

# getfattr -d -m . -e hex /bricks/brick2/b1/f3
getfattr: Removing leading '/' from absolute path names
# file: bricks/brick2/b1/f3
security.selinux=0x73797374656d5f753a6f626a6563745f723a676c75737465726
45f627269636b5f743a733000
trusted.gfid=0x90d9b0f65f6530b95b9f3f8334033df
trusted.gfid2path.364f55367c7bd6f4=0x30303030303030302d303030302d30303
0302d303030302d30303030303030303030312f6634
```

Note

You can not use the GFID of the file as an argument with any of the CLI options to resolve GFID split-brain. It should be the absolute path as seen from the mount point to the file considered as source.

With source-brick option there is no way to resolve all the GFID split-brain in one shot by not specifying any file-path in the CLI as done while resolving data or metadata split-brain. For each file in GFID split-brain, run the CLI with the policy you want to use.

Resolving directory GFID split-brain using CLI with the "source-brick" option in a "distributed-replicated" volume needs to be done on all the volumes explicitly. Since directories get created on all the subvolumes, using one particular brick as source for directory GFID split-brain, heal the directories for that subvolume. In this case, other subvolumes must be healed using the brick which has same GFID as that of the previous brick which was used as source for healing other subvolume. For information on resolving **`entry/type-mismatch`** split-brain, see Chapter 26, *Manually Recovering File Split-brain* .

11.15.4. Triggering Self-Healing on Replicated Volumes

For replicated volumes, when a brick goes offline and comes back online, self-healing is required to re-sync all the replicas. There is a self-heal daemon which runs in the background, and automatically initiates self-healing every 10 minutes on any files which require healing.

Multithreaded Self-heal

Self-heal daemon has the capability to handle multiple heals in parallel and is supported on Replicate and Distribute-replicate volumes. However, increasing the number of heals has impact on I/O performance so the following options have been provided. The **`cluster.shd-max-threads`** volume option controls the number of entries that can be self healed in parallel on each replica by self-heal daemon using. Using **`cluster.shd-wait-qlength`** volume option, you can configure the number of entries that must be kept in the queue for self-heal daemon threads to take up as soon as any of the threads are free to heal.

For more information on **`cluster.shd-max-threads`** and **`cluster.shd-wait-qlength`** volume set options, see Section 11.1, “Configuring Volume Options”.

There are various commands that can be used to check the healing status of volumes and files, or to manually initiate healing:

- To view the list of files that need healing:

  ```
  # gluster volume heal VOLNAME info
  ```

  For example, to view the list of files on test-volume that need healing:

  ```
  # gluster volume heal test-volume info
  Brick server1:/gfs/test-volume_0
  Number of entries: 0

  Brick server2:/gfs/test-volume_1
  /95.txt
  /32.txt
  /66.txt
  /35.txt
  /18.txt
  /26.txt - Possibly undergoing heal
  /47.txt
  /55.txt
  /85.txt - Possibly undergoing heal
  ...
  Number of entries: 101
  ```

- To trigger self-healing only on the files which require healing:

  ```
  # gluster volume heal VOLNAME
  ```

  For example, to trigger self-healing on files which require healing on test-volume:

  ```
  # gluster volume heal test-volume
  Heal operation on volume test-volume has been successful
  ```

- To trigger self-healing on all the files on a volume:

  ```
  # gluster volume heal VOLNAME full
  ```

For example, to trigger self-heal on all the files on test-volume:

```
# gluster volume heal test-volume full
Heal operation on volume test-volume has been successful
```

- To view the list of files on a volume that are in a split-brain state:

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

For example, to view the list of files on test-volume that are in a split-brain state:

```
# gluster volume heal test-volume info split-brain
Brick server1:/gfs/test-volume_2
Number of entries: 12
at                   path on brick
----------------------------------
2012-06-13 04:02:05  /dir/file.83
2012-06-13 04:02:05  /dir/file.28
2012-06-13 04:02:05  /dir/file.69
Brick server2:/gfs/test-volume_2
Number of entries: 12
at                   path on brick
----------------------------------
2012-06-13 04:02:05  /dir/file.83
2012-06-13 04:02:05  /dir/file.28
2012-06-13 04:02:05  /dir/file.69
...
```

11.16. Recommended Configurations - Dispersed Volume

This chapter describes the recommended configurations, examples, and illustrations for Dispersed and Distributed Dispersed volumes.

For a Distributed Dispersed volume, there will be multiple sets of bricks (subvolumes) that stores data with erasure coding. All the files are distributed over these sets of erasure coded subvolumes. In this scenario, even if a redundant number of bricks is lost from every dispersed subvolume, there is no data loss.

For example, assume you have Distributed Dispersed volume of configuration 2 X (4 + 2). Here, you have two sets of dispersed subvolumes where the data is erasure coded between 6 bricks with 2 bricks for redundancy. The files will be stored in one of these dispersed subvolumes. Therefore, even if we lose two bricks from each set, there is no data loss.

Brick Configurations

The following table lists the brick layout details of multiple server/disk configurations for dispersed and distributed dispersed volumes.

Table 11.3. Brick Configurations for Dispersed and Distributed Dispersed Volumes

Redundancy Level	Supported Configurations	Bricks per Server per Subvolume	Node Loss	Max brick failure count within a subvolume	Compatible Server Node count	Increment Size (no. of nodes)	Min number of sub-volumes	Total Spindles	Tolerated HDD Failure Percentage
12 HDD Chassis									
2	4 + 2	2	1	2	3	3	6	36	33.33%
		1	2	2	6	6	12	72	33.33%
2	8+2	2	1	2	5	5	6	60	20.00%
		1	2	2	10	10	12	120	20.00%
3	8 + 3	1-2	1	3	6	6	6	72	25.00%
4	8 + 4	4	1	4	3	3	3	36	33.33%
		2	2	4	6	6	6	72	33.33%
		1	4	4	12	12	12	144	33.33%
4	16 + 4	4	1	4	5	5	3	60	20.00%
		2	2	4	10	10	6	120	20.00%
		1	4	4	20	20	12	240	20.00%
24 HDD Chassis									
2	4 + 2	2	1	2	3	3	12	72	33.33%
		1	2	2	6	6	24	144	33.33%
2	8+ 2	2	1	2	5	5	12	120	20.00%
		1	2	2	10	10	24	240	20.00%
4	8 + 4	4	1	4	3	3	6	72	33.33%
		2	2	4	6	6	12	144	33.33%
		1	4	4	12	12	24	288	33.33%
4	16 + 4	4	1	4	5	5	6	120	20.00%
		2	2	4	10	10	12	240	20.00%
		1	4	4	20	20	24	480	20.00%
36 HDD Chassis									
2	4 + 2	2	1	2	3	3	18	108	33.33%
		1	2	2	6	6	36	216	33.33%
2	8 + 2	2	1	1	5	5	18	180	20.00%
		1	2	2	10	10	36	360	20.00%
3	8 + 3	1-2	1	3	6	6	19	216	26.39%
4	8 + 4	4	1	4	3	3	9	108	33.33%
		2	2	4	6	6	18	216	33.33%
		1	4	4	12	12	36	432	33.33%
4	16 + 4	4	1	4	5	5	9	180	20.00%
		2	2	4	10	10	18	360	20.00%
		1	4	4	20	20	36	720	20.00%
60 HDD Chassis									
2	4 + 2	2	1	2	3	3	30	180	33.33%
		1	2	2	6	6	60	360	33.33%
2	8 + 2	2	1	2	5	5	30	300	20.00%
		1	2	2	10	10	60	600	20.00%
3	8 + 3	1-2	1	3	6	6	32	360	26.67%
4	8 + 4	4	1	4	3	3	15	180	33.33%
		2	2	4	6	6	30	360	33.33%

Redundancy Level	Supported Configurations	Bricks per Server per Subvolume	Node Loss	Max brick failure count within a subvolume	Compatible Server Node count	Increment Size (no. of nodes)	Min number of sub-volumes	Total Spindles	Tolerated HDD Failure Percentage
		1	4	4	12	12	60	720	33.33%
4	16 + 4	4	1	4	5	5	15	300	20.00%
		2	2	4	10	10	30	600	20.00%
		1	4	4	20	20	60	1200	20.00%

Example 1 - Dispersed 4+2 configuration on three servers

This example describes a compact configuration of three servers, with each server attached to a 12 HDD chassis to create a dispersed volume. In this example, each HDD is assumed to contain a single brick.

This example's brick configuration is explained in row 1 of Table 11.3, “Brick Configurations for Dispersed and Distributed Dispersed Volumes”.

With this server-to-spindle ratio, 36 disks/spindles are allocated for the dispersed volume configuration. For example, to create a compact 4+2 dispersed volume using 6 spindles from the total disk pool over three servers, run the following command:

```
# gluster volume create test_vol disperse-data 4 redundancy 2 transport tcp
server1:/rhgs/brick1 server1:/rhgs/brick2 server2:/rhgs/brick3
server2:/rhgs/brick4 server3:/rhgs/brick5 server3:/rhgs/brick6 --force
```

Note that the **`--force`** parameter is required because this configuration is not optimal in terms of fault tolerance. Since each server provides two bricks, this configuration has a greater risk to data availability if a server goes offline than it would if each brick was provided by a separate server.

Run the **`gluster volume info`** command to view the volume information.

```
# gluster volume info test-volume
Volume Name: test-volume
Type: Disperse
Status: Started
Number of Bricks: 1 x (4 + 2) = 6
Transport-type: tcp
Bricks:
Brick1: server1:/rhgs/brick1
Brick2: server1:/rhgs/brick2
Brick3: server2:/rhgs/brick3
Brick4: server2:/rhgs/brick4
Brick5: server3:/rhgs/brick5
Brick6: server3:/rhgs/brick6
```

Additionally, you can convert the dispersed volume to a distributed dispersed volume in increments of 4+2. Add six bricks from the disk pool using the following command:

```
# gluster volume add-brick test_vol server1:/rhgs/brick7 server1:/rhgs/brick8
server2:/rhgs/brick9 server2:/rhgs/brick10 server3:/rhgs/brick11
server3:/rhgs/brick12
```

Run the **`gluster volume info`** command to view distributed dispersed volume information.

```
# gluster volume info test-volume
Volume Name: test-volume
Type: Distributed-Disperse
Status: Started
Number of Bricks: 2 x (4 + 2) = 12
Transport-type: tcp
Bricks:
Brick1: server1:/rhgs/brick1
Brick2: server1:/rhgs/brick2
Brick3: server2:/rhgs/brick3
Brick4: server2:/rhgs/brick4
Brick5: server3:/rhgs/brick5
Brick6: server3:/rhgs/brick6
Brick7: server1:/rhgs/brick7
Brick8: server1:/rhgs/brick8
Brick9: server2:/rhgs/brick9
Brick10: server2:/rhgs/brick10
Brick11: server3:/rhgs/brick11
Brick12: server3:/rhgs/brick12
```

Using this configuration example, you can create configuration combinations of 6 x (4 + 2) distributed dispersed volumes. This example configuration has tolerance up to 12 brick failures.

For details about creating an optimal configuration, see Section 5.9, “Creating Dispersed Volumes”.

Example 2 - Dispersed 8+4 configuration on three servers

The following diagram illustrates a dispersed 8+4 configuration on three servers as explained in the row 3 of Table 11.3, “Brick Configurations for Dispersed and Distributed Dispersed Volumes” The command to create the disperse volume for this configuration:

```
# gluster volume create test_vol disperse-data 8 redundancy 4 transport tcp
server1:/rhgs/brick1 server1:/rhgs/brick2 server1:/rhgs/brick3
server1:/rhgs/brick4 server2:/rhgs/brick1 server2:/rhgs/brick2
server2:/rhgs/brick3 server2:/rhgs/brick4 server3:/rhgs/brick1
server3:/rhgs/brick2 server3:/rhgs/brick3 server3:/rhgs/brick4
server1:/rhgs/brick5 server1:/rhgs/brick6 server1:/rhgs/brick7
server1:/rhgs/brick8 server2:/rhgs/brick5 server2:/rhgs/brick6
server2:/rhgs/brick7 server2:/rhgs/brick8  server3:/rhgs/brick5
server3:/rhgs/brick6 server3:/rhgs/brick7 server3:/rhgs/brick8
server1:/rhgs/brick9 server1:/rhgs/brick10 server1:/rhgs/brick11
server1:/rhgs/brick12 server2:/rhgs/brick9 server2:/rhgs/brick10
server2:/rhgs/brick11 server2:/rhgs/brick12 server3:/rhgs/brick9
server3:/rhgs/brick10 server3:/rhgs/brick11 server3:/rhgs/brick12 --force
```

Note that the **`--force`** parameter is required because this configuration is not optimal in terms of fault tolerance. Since each server provides more than one brick, this configuration has a greater risk to data availability if a server goes offline than it would if each brick was provided by a separate server.

For details about creating an optimal configuration, see Section 5.9, “Creating Dispersed Volumes”.

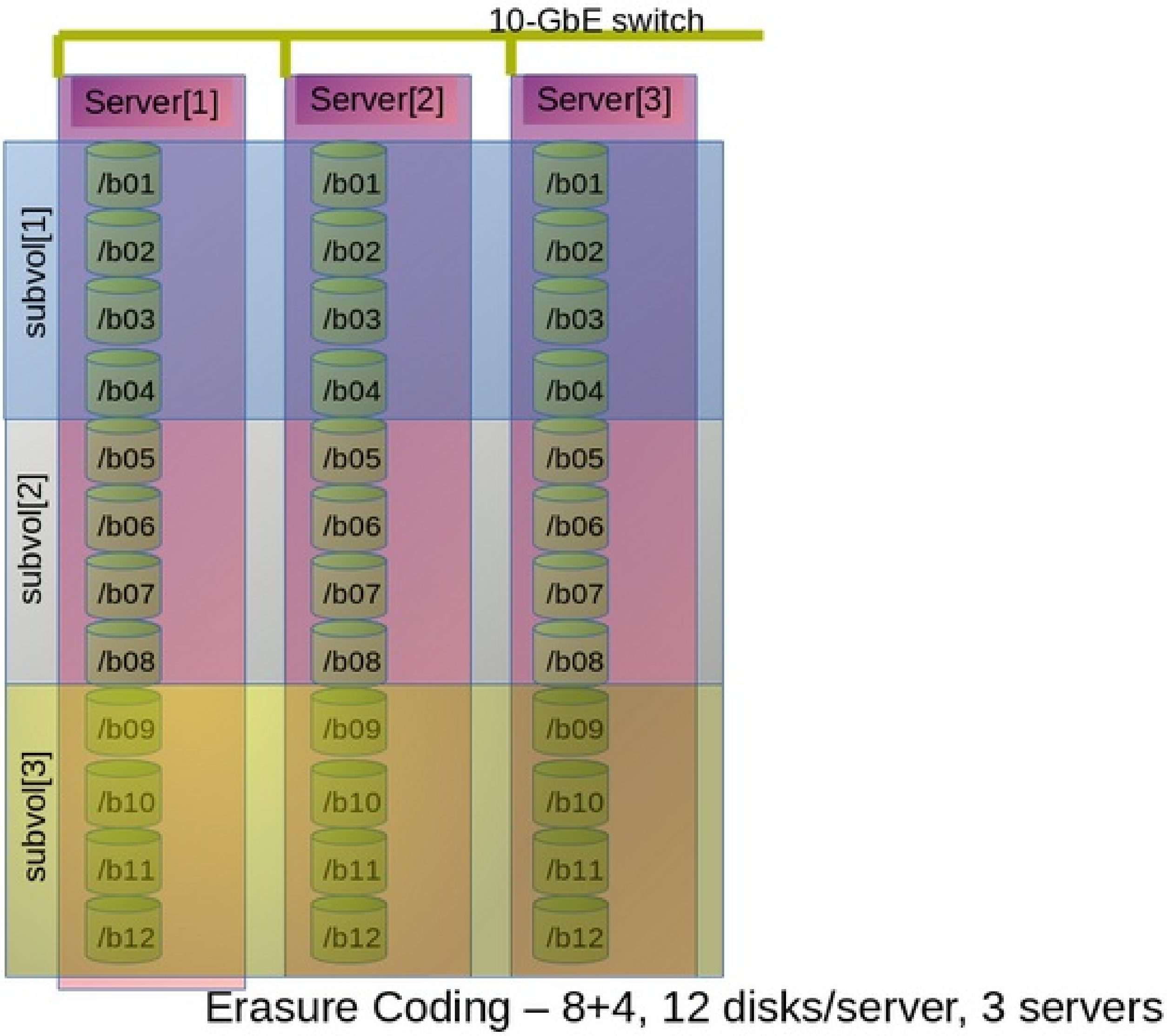


Figure 11.1. Example Configuration of 8+4 Dispersed Volume Configuration

In this example, there are `m` bricks (refer to section Section 5.9, "Creating Dispersed Volumes" for information on **`n = k+m`** equation) from a dispersed subvolume on each server. If you add more than `m` bricks from a dispersed subvolume on server **`S,`** and if the server **`S`** goes down, data will be unavailable.

If **`S`** (a single column in the above diagram) goes down, there is no data loss, but if there is any additional hardware failure, either another node going down or a storage device failure, there would be immediate data loss.

Example 3 - Dispersed 4+2 configuration on six servers

The following diagram illustrates dispersed 4+2 configuration on six servers and each server with 12-disk-per-server configuration as explained in the row 2 of Table 11.3, "Brick Configurations for Dispersed and Distributed Dispersed Volumes". The command to create the disperse volume for this configuration:

```
# gluster volume create test_vol disperse-data 4 redundancy 2 transport tcp
server1:/rhgs/brick1 server2:/rhgs/brick1 server3:/rhgs/brick1
server4:/rhgs/brick1 server5:/rhgs/brick1
server6:/rhgs/brick1server1:/rhgs/brick2 server2:/rhgs/brick2
server3:/rhgs/brick2 server4:/rhgs/brick2 server5:/rhgs/brick2
server6:/rhgs/brick2 server1:/rhgs/brick3 server2:/rhgs/brick3
server3:/rhgs/brick3 server4:/rhgs/brick3 server5:/rhgs/brick3
```

```
server6:/rhgs/brick3 server1:/rhgs/brick4 server2:/rhgs/brick4
server3:/rhgs/brick4 server4:/rhgs/brick4 server5:/rhgs/brick4
server6:/rhgs/brick4 server1:/rhgs/brick5 server2:/rhgs/brick5
server3:/rhgs/brick5 server4:/rhgs/brick5 server5:/rhgs/brick5
server6:/rhgs/brick5 server1:/rhgs/brick6 server2:/rhgs/brick6
server3:/rhgs/brick6 server4:/rhgs/brick6 server5:/rhgs/brick6
server6:/rhgs/brick6 server1:/rhgs/brick7 server2:/rhgs/brick7
server3:/rhgs/brick7 server4:/rhgs/brick7 server5:/rhgs/brick7
server6:/rhgs/brick7 server1:/rhgs/brick8 server2:/rhgs/brick8
server3:/rhgs/brick8 server4:/rhgs/brick8 server5:/rhgs/brick8
server6:/rhgs/brick8 server1:/rhgs/brick9 server2:/rhgs/brick9
server3:/rhgs/brick9 server4:/rhgs/brick9 server5:/rhgs/brick9
server6:/rhgs/brick9 server1:/rhgs/brick10 server2:/rhgs/brick10
server3:/rhgs/brick10 server4:/rhgs/brick10 server5:/rhgs/brick10
server6:/rhgs/brick10 server1:/rhgs/brick11 server2:/rhgs/brick11
server3:/rhgs/brick11 server4:/rhgs/brick11 server5:/rhgs/brick11
server6:/rhgs/brick11 server1:/rhgs/brick12 server2:/rhgs/brick12
server3:/rhgs/brick12 server4:/rhgs/brick12 server5:/rhgs/brick12
server6:/rhgs/brick12
```

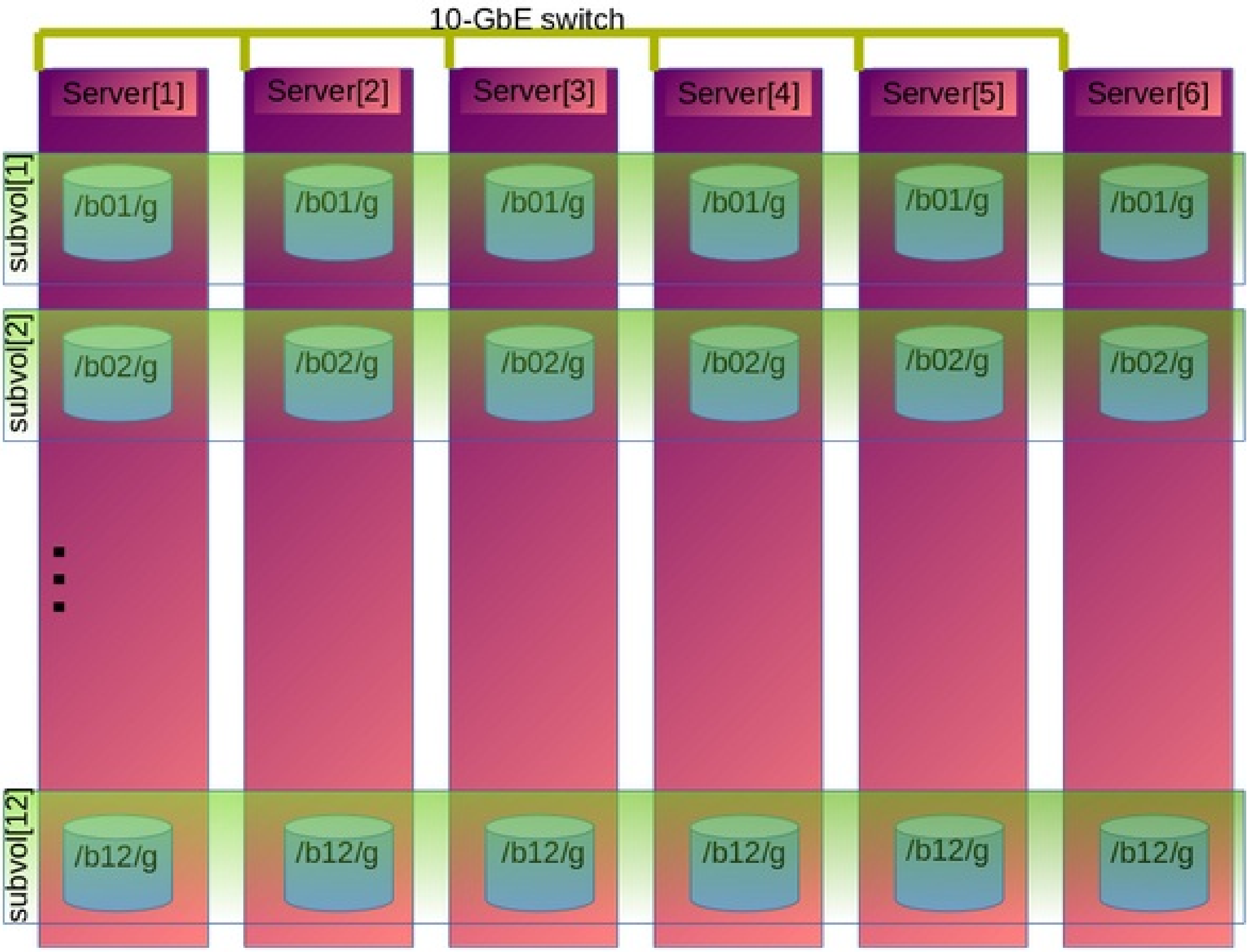


Figure 11.2. Example Configuration of 4+2 Dispersed Volume Configuration

Redundancy Comparison

The following chart illustrates the redundancy comparison of all supported dispersed volume configurations.

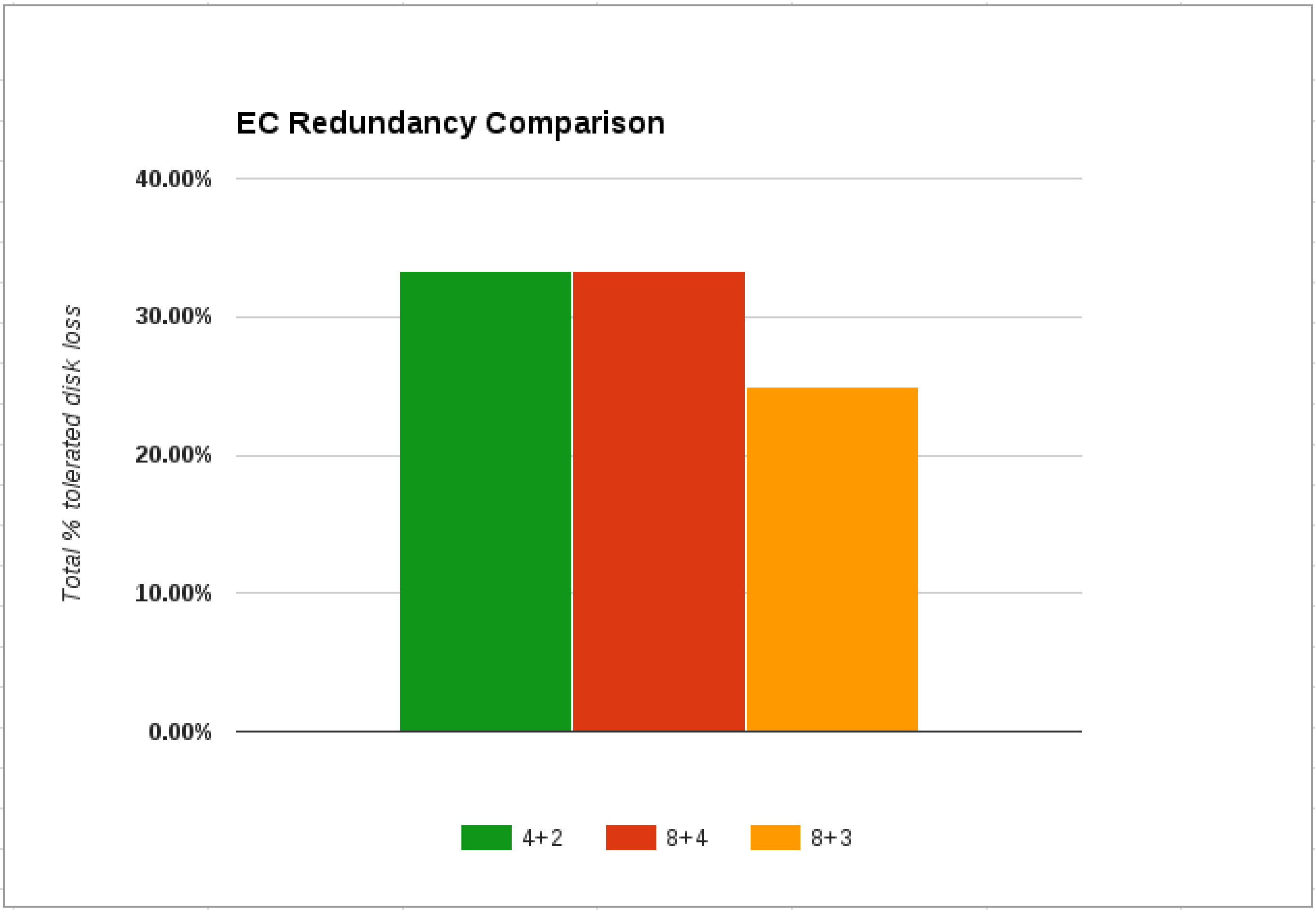


Figure 11.3. Illustration of the redundancy comparison

Chapter 12. Managing Red Hat Gluster Storage Logs

The log management framework generates log messages for each of the administrative functionalities and the components to increase the user-serviceability aspect of Red Hat Gluster Storage Server. Logs are generated to track the event changes in the system. The feature makes the retrieval, rollover, and archival of log files easier and helps in troubleshooting errors that are user-resolvable with the help of the Red Hat Gluster Storage Error Message Guide. The Red Hat Gluster Storage Component logs are rotated on a weekly basis. Administrators can rotate a log file in a volume, as needed. When a log file is rotated, the contents of the current log file are moved to **`log-file-name.epoch-time-stamp`**.The components for which the log messages are generated with message-ids are glusterFS Management Service, Distributed Hash Table (DHT), and Automatic File Replication (AFR).

12.1. Log Rotation

Log files are rotated on a weekly basis and the log files are zipped in the gzip format on a fortnightly basis. When the content of the log file is rotated, the current log file is moved to log-file- name.epoch-time-stamp. The archival of the log files is defined in the configuration file. As a policy, log file content worth 52 weeks is retained in the Red Hat Gluster Storage Server.

12.2. Red Hat Gluster Storage Component Logs and Location

The table lists the component, services, and functionality based logs in the Red Hat Gluster Storage Server. As per the File System Hierarchy Standards (FHS) all the log files are placed in the **`/var/log`** directory.

Table 12.1.

Component/Service Name	Location of the Log File	Remarks
glusterd	**`/var/log/glusterfs/glusterd.log`**	One glusterd log file per server. This log file also contains the snapshot and user logs.
gluster commands	**`/var/log/glusterfs/cmd_history.log`**	Gluster commands executed on a node in a Red Hat Gluster Storage Trusted Storage Pool is logged in this file.
bricks	**`/var/log/glusterfs/bricks/`*`<path extraction of brick path>`*`.log`**	One log file per brick on the server
rebalance	**`/var/log/glusterfs/`*`VOLNAME`*`-rebalance.log`**	One log file per volume on the server
self heal deamon	**`/var/log/glusterfs/glustershd.log`**	One log file per server

Component/Service Name	Location of the Log File	Remarks
quota	- **/var/log/glusterfs/quotad.log** Log of the quota daemons running on each node. - **/var/log/glusterfs/quota-crawl.log** Whenever quota is enabled, a file system crawl is performed and the corresponding log is stored in this file - **/var/log/glusterfs/quota-mount-*VOLNAME*.log** An auxiliary FUSE client is mounted in <gluster-run-dir>/*VOLNAME* of the glusterFS and the corresponding client logs found in this file.	One log file per server (and per volume from quota-mount.
Gluster NFS	**/var/log/glusterfs/nfs.log**	One log file per server
SAMBA Gluster	**/var/log/samba/glusterfs-*VOLNAME*-<ClientIP>.log**	If the client mounts this on a glusterFS server node, the actual log file or the mount point may not be found. In such a case, the mount outputs of all the glusterFS type mount operations need to be considered.
NFS - Ganesha	**/var/log/ganesha/ganesha.log**, **/var/log/ganesha/ganesha-gfapi.log**	One log file per server
FUSE Mount	**/var/log/glusterfs/<*mountpoint path extraction*>.log**	
Geo-replication	**/var/log/glusterfs/geo-replication/<master>** **/var/log/glusterfs/geo-replication-slaves**	
gluster volume heal *VOLNAME* info command	**/var/log/glusterfs/glfsheal-*VOLNAME*.log**	One log file per server on which the command is executed.
gluster-swift	**/var/log/messages**	
SwiftKrbAuth	**/var/log/httpd/error_log**	
Command Line Interface logs	**/var/log/glusterfs/*cli*.log**	This file captures log entries for every command that is executed on the Command Line Interface(CLI).

12.3. Configuring the Log Format

You can configure the Red Hat Gluster Storage Server to generate log messages either with message IDs or without them.

To know more about these options, see topic *Configuring Volume Options* in the *Red Hat Gluster Storage Administration Guide*.

To configure the log-format for bricks of a volume:

```
# gluster volume set VOLNAME diagnostics.brick-log-format <value>
```

Example 12.1. Generate log files with `with-msg-id`:

```
# gluster volume set testvol diagnostics.brick-log-format with-msg-id
```

Example 12.2. Generate log files with `no-msg-id`:

```
# gluster volume set testvol diagnostics.brick-log-format no-msg-id
```

To configure the log-format for clients of a volume:

```
gluster volume set VOLNAME diagnostics.client-log-format <value>
```

Example 12.3. Generate log files with `with-msg-id`:

```
# gluster volume set testvol diagnostics.client-log-format with-msg-id
```

Example 12.4. Generate log files with `no-msg-id`:

```
# gluster volume set testvol diagnostics.client-log-format no-msg-id
```

To configure the log format for `glusterd`:

```
# glusterd --log-format=<value>
```

Example 12.5. Generate log files with `with-msg-id`:

```
# glusterd --log-format=with-msg-id
```

Example 12.6. Generate log files with `no-msg-id`:

```
# glusterd --log-format=no-msg-id
```

To a list of error messages, see the *Red Hat Gluster Storage Error Message Guide*.

See Also:

- Section 11.1, “Configuring Volume Options”

12.4. Configuring the Log Level

Every log message has a log level associated with it. The levels, in descending order, are CRITICAL, ERROR, WARNING, INFO, DEBUG, and TRACE. Red Hat Gluster Storage can be configured to generate log messages only for certain log levels. Only those messages that have log levels above or equal to the configured log level are logged.

For example, if the log level is set to **`INFO`**, only **`CRITICAL`**, **`ERROR`**, **`WARNING`**, and **`INFO`** messages are logged.

The components can be configured to log at one of the following levels:

- CRITICAL
- ERROR
- WARNING
- INFO
- DEBUG
- TRACE

Important

Setting the log level to TRACE or DEBUG generates a very large number of log messages and can lead to disks running out of space very quickly.

To configure the log level on bricks

```
# gluster volume set VOLNAME diagnostics.brick-log-level <value>
```

Example 12.7. Set the log level to warning on a brick

```
# gluster volume set testvol diagnostics.brick-log-level WARNING
```

To configure the syslog level on bricks

```
# gluster volume set VOLNAME diagnostics.brick-sys-log-level <value>
```

Example 12.8. Set the syslog level to warning on a brick

```
# gluster volume set testvol diagnostics.brick-sys-log-level WARNING
```

To configure the log level on clients

```
# gluster volume set VOLNAME diagnostics.client-log-level <value>
```

Example 12.9. Set the log level to error on a client

```
# gluster volume set testvol diagnostics.client-log-level ERROR
```

To configure the syslog level on clients

```
# gluster volume set VOLNAME diagnostics.client-sys-log-level <value>
```

Example 12.10. Set the syslog level to error on a client

```
# gluster volume set testvol diagnostics.client-sys-log-level ERROR
```

To configure the log level forglusterd persistently

Edit the **/etc/sysconfig/glusterd** file, and set the value of the ***LOG_LEVEL*** parameter to the log level that you want glusterd to use.

```
## Set custom log file and log level (below are defaults)
#LOG_FILE='/var/log/glusterfs/glusterd.log'
LOG_LEVEL='VALUE'
```

This change does not take effect until glusterd is started or restarted with the **service** or **systemctl** command.

Example 12.11. Set the log level to WARNING onglusterd

In the **/etc/sysconfig/glusterd** file, locate the ***LOG_LEVEL*** parameter and set its value to **WARNING**.

```
## Set custom log file and log level (below are defaults)
#LOG_FILE='/var/log/glusterfs/glusterd.log'
LOG_LEVEL='WARNING'
```

Then start or restart the glusterd service. On Red Hat Enterprise Linux 7, run:

```
# systemctl restart glusterd.service
```

On Red Hat Enterprise Linux 6, run:

```
# service glusterd restart
```

To run a gluster command once with a specified log level

```
# gluster --log-level=ERROR VOLNAME COMMAND
```

Example 12.12. Run `volume status` with a log level of ERROR

```
# gluster --log-level=ERROR volume status
```

See Also:

- Section 11.1, “Configuring Volume Options”

12.5. Suppressing Repetitive Log Messages

Repetitive log messages in the Red Hat Gluster Storage Server can be configured by setting a **`log-flush-timeout`** period and by defining a **`log-buf-size`** buffer size options with the **`gluster volume set`** command.

Suppressing Repetitive Log Messages with a Timeout Period

To set the timeout period on the bricks:

```
# gluster volume set VOLNAME diagnostics.brick-log-flush-timeout <value>
```

Example 12.13. Set a timeout period on the bricks

```
# gluster volume set testvol diagnostics.brick-log-flush-timeout 200
volume set: success
```

To set the timeout period on the clients:

```
# gluster volume set VOLNAME diagnostics.client-log-flush-timeout <value>
```

Example 12.14. Set a timeout period on the clients

```
# gluster volume set testvol diagnostics.client-log-flush-timeout 180
volume set: success
```

To set the timeout period on `glusterd`:

```
# glusterd --log-flush-timeout=<value>
```

Example 12.15. Set a timeout period on the `glusterd`

```
# glusterd --log-flush-timeout=60
```

Suppressing Repetitive Log Messages by defining a Buffer Size

The maximum number of unique log messages that can be suppressed until the timeout or buffer overflow, whichever occurs first on the bricks.

To set the buffer size on the bricks:

```
# gluster volume set VOLNAME diagnostics.brick-log-buf-size <value>
```

Example 12.16. Set a buffer size on the bricks

```
# gluster volume set testvol diagnostics.brick-log-buf-size 10
volume set: success
```

To set the buffer size on the clients:

```
# gluster volume set VOLNAME diagnostics.client-log-buf-size <value>
```

Example 12.17. Set a buffer size on the clients

```
# gluster volume set testvol diagnostics.client-log-buf-size 15
volume set: success
```

To set the log buffer size on `glusterd`:

```
# glusterd --log-buf-size=<value>
```

Example 12.18. Set a log buffer size on the `glusterd`

```
# glusterd --log-buf-size=10
```

Note

To disable suppression of repetitive log messages, set the log-buf-size to zero.

See Also:

- Section 11.1, “Configuring Volume Options”

12.6. Geo-replication Logs

The following log files are used for a geo-replication session:

- **`Master-log-file`** - log file for the process that monitors the master volume.
- **`Slave-log-file`** - log file for process that initiates changes on a slave.

- **Master-gluster-log-file** - log file for the maintenance mount point that the geo-replication module uses to monitor the master volume.

- **Slave-gluster-log-file** - If the slave is a Red Hat Gluster Storage Volume, this log file is the slave's counterpart of **Master-gluster-log-file**.

12.6.1. Viewing the Geo-replication Master Log Files

To view the Master-log-file for geo-replication, use the following command:

```
# gluster volume geo-replication MASTER_VOL SLAVE_HOST::SLAVE_VOL config
log-file
```

For example:

```
# gluster volume geo-replication Volume1 example.com::slave-vol config log-
file
```

12.6.2. Viewing the Geo-replication Slave Log Files

To view the log file for geo-replication on a slave, use the following procedure. **glusterd** must be running on slave machine.

1. On the master, run the following command to display the session-owner details:

   ```
   # gluster volume geo-replication MASTER_VOL SLAVE_HOST::SLAVE_VOL
   config session-owner
   ```

   For example:

   ```
   # gluster volume geo-replication Volume1 example.com::slave-vol config
   session-owner 5f6e5200-756f-11e0-a1f0-0800200c9a66
   ```

2. On the slave, run the following command with the session-owner value from the previous step:

   ```
   # gluster volume geo-replication SLAVE_VOL config log-file
   /var/log/gluster/SESSION_OWNER:remote-mirror.log
   ```

   For example:

   ```
   # gluster volume geo-replication slave-vol config log-file
   /var/log/gluster/5f6e5200-756f-11e0-a1f0-0800200c9a66:remote-
   mirror.log
   ```

Chapter 13. Managing Red Hat Gluster Storage Volume Life-Cycle Extensions

Red Hat Gluster Storage allows automation of operations by user-written scripts. For every operation, you can execute a pre and a post script.

Pre Scripts: These scripts are run before the occurrence of the event. You can write a script to automate activities like managing system-wide services. For example, you can write a script to stop exporting the SMB share corresponding to the volume before you stop the volume.

Post Scripts: These scripts are run after execution of the event. For example, you can write a script to export the SMB share corresponding to the volume after you start the volume.

You can run scripts for the following events:

- Creating a volume
- Starting a volume
- Adding a brick
- Removing a brick
- Tuning volume options
- Stopping a volume
- Deleting a volume

Naming Convention

While creating the file names of your scripts, you must follow the naming convention followed in your underlying file system like XFS.

> **Note**
>
> To enable the script, the name of the script must start with an **S** . Scripts run in lexicographic order of their names.

13.1. Location of Scripts

This section provides information on the folders where the scripts must be placed. When you create a trusted storage pool, the following directories are created:

- /var/lib/glusterd/hooks/1/create/
- /var/lib/glusterd/hooks/1/delete/
- /var/lib/glusterd/hooks/1/start/
- /var/lib/glusterd/hooks/1/stop/
- /var/lib/glusterd/hooks/1/set/
- /var/lib/glusterd/hooks/1/add-brick/

- /var/lib/glusterd/hooks/1/remove-brick/

After creating a script, you must ensure to save the script in its respective folder on all the nodes of the trusted storage pool. The location of the script dictates whether the script must be executed before or after an event. Scripts are provided with the command line argument **`--volname=`*`VOLNAME`*** to specify the volume. Command-specific additional arguments are provided for the following volume operations:

- Start volume
  - **`--first=yes`**, if the volume is the first to be started
  - **`--first=no`**, for otherwise
- Stop volume
  - **`--last=yes`**, if the volume is to be stopped last.
  - **`--last=no`**, for otherwise
- Set volume
  - **`-o key=value`**

    For every key, value is specified in volume set command.

13.2. Prepackaged Scripts

Red Hat provides scripts to export Samba (SMB) share when you start a volume and to remove the share when you stop the volume. These scripts are available at: **`/var/lib/glusterd/hooks/1/start/post`** and **`/var/lib/glusterd/hooks/1/stop/pre`**. By default, the scripts are enabled.

When you start a volume using the following command:

```
# gluster volume start VOLNAME
```

The **`S30samba-start.sh`** script performs the following:

1. Adds Samba share configuration details of the volume to the **`smb.conf`** file
2. Mounts the volume through FUSE and adds an entry in **`/etc/fstab`** for the same.
3. Restarts Samba to run with updated configuration

When you stop the volume using the following command:

```
# gluster volume stop VOLNAME
```

The **`S30samba-stop.sh`** script performs the following:

1. Removes the Samba share details of the volume from the **`smb.conf`** file
2. Unmounts the FUSE mount point and removes the corresponding entry in **`/etc/fstab`**
3. Restarts Samba to run with updated configuration

Chapter 14. Managing Containerized Red Hat Gluster Storage

Red Hat Gluster Storage can be set up as a container on a Red Hat Enterprise Linux Atomic Host. Containers use the shared kernel concept and are much more efficient than hypervisors in system resource terms. Containers rest on top of a single Linux instance and allows applications to use the same Linux kernel as the system that they are running on. This improves the overall efficiency and reduces the space consumption considerably.

Containerized Red Hat Gluster Storage 3.1.2 is supported only on Red Hat Enterprise Linux Atomic Host 7.2. For more information about installing containerized Red Hat Gluster Storage, see the Red Hat Gluster Storage 3.4 *Installation Guide*: https://access.redhat.com/documentation/en-us/red_hat_gluster_storage/3.4/html/installation_guide/.

Note

For Red Hat Gluster Storage 3.1.2, Erasure Coding, NFS-Ganesha, BitRot, and Data Tiering are not supported with containerized Red Hat Gluster Storage.

14.1. Prerequisites

Before creating a container, execute the following steps.

1. Create the directories in the atomic host for persistent mount by executing the following command:

   ```
   # mkdir -p /etc/glusterfs /var/lib/glusterd /var/log/glusterfs
   ```

2. Ensure the bricks that are required are mounted on the atomic hosts. For more information see Section 21.2, “Brick Configuration”.

3. If Snapshot is required, then ensure that the **dm-snapshot** kernel module is loaded in Atomic Host system. If it is not loaded, then load it by executing the following command:

   ```
   # modprobe dm_snapshot
   ```

14.2. Starting a Container

Execute the following steps to start the container.

1. Execute the following command to run the container:

   ```
   # docker run -d --privileged=true --net=host --name <container-name> -
   v /run -v /etc/glusterfs:/etc/glusterfs:z -v
   /var/lib/glusterd:/var/lib/glusterd:z -v
   /var/log/glusterfs:/var/log/glusterfs:z -v
   /sys/fs/cgroup:/sys/fs/cgroup:ro -v
   /mnt/brick1:/mnt/container_brick1:z <image name>
   ```

   where,

   - --net=host option ensures that the container has full access to the network stack of the host.

- **/mnt/brick1** is the mountpoint of the brick in the atomic host and **:/mnt/container_brick1** is the mountpoint of the brick in the container.

- -d option starts the container in the detached mode.

For example:

```
# docker run -d --privileged=true --net=host --name glusternode1 -v
/run -v /etc/glusterfs:/etc/glusterfs:z -v
/var/lib/glusterd:/var/lib/glusterd:z -v
/var/log/glusterfs:/var/log/glusterfs:z -v
/sys/fs/cgroup:/sys/fs/cgroup:ro -v
/mnt/brick1:/mnt/container_brick1:z rhgs3/rhgs-server-rhel7

5ac864b5abc74a925aecc4fe9613c73e83b8c54a846c36107aa8e2960eeb97b4
```

Where, 5ac864b5abc74a925aecc4fe9613c73e83b8c54a846c36107aa8e2960eeb97b4 is the container ID.

Note

- SELinux labels are automatically reset to **svirt_sandbox_file_t** so that the container can interact with the Atomic Host directory.
- In the above command, the following ensures that the gluster configuration are persistent.

```
-v /etc/glusterfs:/etc/glusterfs:z -v
/var/lib/glusterd:/var/lib/glusterd -v
/var/log/glusterfs:/var/log/glusterfs
```

2. If you want to use snapshot then execute the following command:

```
# docker run -d --privileged=true --net=host --name <container-name> -
v /dev:/dev -v /run -v /etc/glusterfs:/etc/glusterfs:z -v
/var/lib/glusterd:/var/lib/glusterd:z -v
/var/log/glusterfs:/var/log/glusterfs:z -v
/sys/fs/cgroup:/sys/fs/cgroup:ro -v
/mnt/brick1:/mnt/container_brick1:z <image name>
```

where, /mnt/brick1 is the mountpoint of the brick in the atomic host and :/mnt/container_brick1 is the mountpoint of the brick in the container.

For example:

```
# docker run -d --privileged=true --net=host --name glusternode1 -v
/dev:/dev -v /run -v /etc/glusterfs:/etc/glusterfs:z -v
/var/lib/glusterd:/var/lib/glusterd:z -v
/var/log/glusterfs:/var/log/glusterfs:z -v
/sys/fs/cgroup:/sys/fs/cgroup:ro -v
/mnt/brick1:/mnt/container_brick1:z rhgs3/rhgs-server-rhel7

5da2bc217c0852d2b1bfe4fb31e0181753410071584b4e38bd77d7502cd3e92b
```

3. To verify if the container is created, execute the following command:

   ```
   # docker ps
   ```

   For example:

   ```
   # docker ps

   CONTAINER ID        IMAGE               COMMAND
   CREATED             STATUS                      PORTS
   NAMES
   5da2bc217c08        891ea0584e94        "/usr/sbin/init"       10
   seconds ago      Up 9 seconds
   glusternode1
   ```

14.3. Creating a Trusted Storage Pool

Perform the following steps to create a Trusted Storage Pool:

1. Access the container using the following command:

   ```
   # docker exec -it <container-name> /bin/bash
   ```

   For example:

   ```
   # docker exec -it glusternode1 /bin/bash
   ```

2. To verify if glusterd is running, execute the following command:

   ```
   # systemctl status glusterd
   ```

3. To verify if the bricks are mounted successfully, execute the following command:

   ```
   # mount |grep <brick_name>
   ```

4. Peer probe the container to form the Trusted Storage Pool:

   ```
   # gluster peer probe <atomic host IP>
   ```

5. Execute the following command to verify the peer probe status:

   ```
   # gluster peer status
   ```

14.4. Creating a Volume

Perform the following steps to create a volume.

1. To create a volume execute the following command:

   ```
   # gluster volume create <vol-name> IP:/brickpath
   ```

2. Start the volume by executing the following command:

   ```
   # gluster volume start <volname>
   ```

14.5. Mounting a Volume

Execute the following command to mount the volume created earlier:

```
# mount -t glusterfs <atomic host IP>:/<vol-name> /mount/point
```

Chapter 15. Detecting BitRot

BitRot detection is a technique used in Red Hat Gluster Storage to identify when silent corruption of data has occurred. BitRot also helps to identify when a brick's data has been manipulated directly, without using FUSE, NFS or any other access protocols. BitRot detection is particularly useful when using JBOD, since JBOD does not provide other methods of determining when data on a disk has become corrupt.

The **`gluster volume bitrot`** command scans all the bricks in a volume for BitRot issues in a process known as scrubbing. The process calculates the checksum for each file or object, and compares that checksum against the actual data of the file. When BitRot is detected in a file, that file is marked as corrupted, and the detected errors are logged in the following files:

- /var/log/glusterfs/bitd.log
- /var/log/glusterfs/scrub.log

15.1. Enabling and Disabling the BitRot daemon

The BitRot daemon is disabled by default. In order to use or configure the daemon, you first need to enable it.

gluster volume bitrot *VOLNAME* enable

Enable the BitRot daemon for the specified volume.

gluster volume bitrot *VOLNAME* disable

Disable the BitRot daemon for the specified volume.

15.2. Modifying BitRot Detection Behavior

Once the daemon is enabled, you can pause and resume the detection process, check its status, and modify how often or how quickly it runs.

gluster volume bitrot *VOLNAME* scrub ondemand

Starts the scrubbing process and the scrubber will start crawling the file system immediately. Ensure to keep the scrubber in 'Active (Idle)' state, where the scrubber is waiting for it's next frequency cycle to start scrubbing, for on demand scrubbing to be successful. On demand scrubbing does not work when the scrubber is in 'Paused' state or already running.

gluster volume bitrot *VOLNAME* scrub pause

Pauses the scrubbing process on the specified volume. Note that this does not stop the BitRot daemon; it stops the process that cycles through the volume checking files.

gluster volume bitrot *VOLNAME* scrub resume

Resumes the scrubbing process on the specified volume. Note that this does not start the BitRot daemon; it restarts the process that cycles through the volume checking files.

gluster volume bitrot *VOLNAME* scrub status

This command prints a summary of scrub status on the specified volume, including various configuration details and the location of the bitrot and scrubber error logs for this volume. It also prints details each node scanned for errors, along with identifiers for any corrupted objects located.

gluster volume bitrot *VOLNAME* scrub-throttle *rate*

Because the BitRot daemon scrubs the entire file system, scrubbing can have a severe performance impact. This command changes the rate at which files and objects are verified. Valid rates are **lazy**, **normal**, and **aggressive**. By default, the scrubber process is started in **lazy** mode.

gluster volume bitrot *VOLNAME* scrub-frequency *frequency*

This command changes how often the scrub operation runs when the BitRot daemon is enabled. Valid options are **daily**, **weekly**, **biweekly**, and **monthly**.By default, the scrubber process is set to run **biweekly**.

15.3. Restoring a bad file

When bad files are revealed by the scrubber, you can perform the following process to heal the file by recovering a copy from a replicate volume.

Important

The following procedure is easier if GFID-to-path translation is enabled.

Mount all volumes using the **-oaux-gfid-mount** mount option, and enable GFID-to-path translation on each volume by running the following command.

```
# gluster volume set VOLNAME build-pgfid on
```

Files created before this option was enabled must be looked up with the **find** command.

Procedure 15.1. Restoring a bad file from a replicate volume

1. **Note the identifiers of bad files**

   Check the output of the **scrub status** command to determine the identifiers of corrupted files.

   ```
   # gluster volume bitrot VOLNAME scrub status
   Volume name: VOLNAME
   ...
   Node name: NODENAME
   ...
   Error count: 3
   Corrupted objects:
   5f61ade8-49fb-4c37-af84-c95041ff4bf5
   e8561c6b-f881-499b-808b-7fa2bce190f7
   eff2433f-eae9-48ba-bdef-839603c9434c
   ```

2. **Determine the path of each corrupted object**

   For files created after GFID-to-path translation was enabled, use the **getfattr** command to determine the path of the corrupted files.

```
# getfattr -n glusterfs.ancestry.path -e text
/mnt/VOLNAME/.gfid/GFID
...
glusterfs.ancestry.path="/path/to/corrupted_file"
```

For files created before GFID-to-path translation was enabled, use the **find** command to determine the path of the corrupted file and the index file that match the identifying GFID.

```
# find /rhgs/brick*/.glusterfs -name GFID
/rhgs/brick1/.glusterfs/path/to/GFID
```

```
# find /rhgs -samefile /rhgs/brick1/.glusterfs/path/to/GFID
/rhgs/brick1/.glusterfs/path/to/GFID
/rhgs/brick1/path/to/corrupted_file
```

3. **Delete the corrupted files**

   Delete the corrupted files from the path output by the **getfattr** or **find** command.

4. **Delete the GFID file**

   Delete the GFID file from the ***/rhgs/brickN/.glusterfs*** directory.

5. **Restore the file**

   Follow these steps to safely restore corrupt files.

   a. **Disable metadata caching**

   If the metadata cache is enabled, disable it by running the following command:

   ```
   # gluster volume set VOLNAME stat-prefetch off
   ```

   b. **Create a recovery mount point**

   Create a mount point to use for the recovery process. For example, **/mnt/recovery**.

   ```
   # mkdir /mnt/recovery
   ```

   c. **Mount the volume with timeouts disabled**

   ```
   # mount -t glusterfs -o attribute-timeout=0,entry-timeout=0
   hostname:volume-path /mnt/recovery
   ```

   d. **Heal files and hard links**

   Access files and hard links to heal them. For example, run the **stat** command on the files and hard links you need to heal.

   ```
   $ stat /mnt/recovery/corrupt-file
   ```

If you do not have client self-heal enabled, you must manually heal the volume with the following command.

```
# gluster volume heal VOLNAME
```

e. **Unmount and optionally remove the recovery mount point**

```
# umount /mnt/recovery
# rmdir /mnt/recovery
```

f. **Optional: Re-enable metadata caching**

If the metadata cache was enabled previously, re-enable it by running the following command:

```
# gluster volume set VOLNAME stat-prefetch on
```

The next time that the bitrot scrubber runs, this GFID is no longer listed (unless it has become corrupted again).

Chapter 16. Incremental Backup Assistance using Glusterfind

Glusterfind is a utility that provides the list of files that are modified between the previous backup session and the current period. The commands can be executed at regular intervals to retrieve the list. Multiple sessions for the same volume can be present for different use cases. The changes that are recorded are, new file/directories, data/metadata modifications, rename, and deletes.

16.1. Glusterfind Configuration Options

The following is the list configuration options available in Glusterfind:

- Glusterfind Create
- Glusterfind Pre
- Glusterfind Post
- Glusterfind Query
- Glusterfind List
- Glusterfind Delete

Note

All the glusterfind configuration commands such as, glusterfind pre, glusterfind post, glusterfind list, and glusterfind delete for a session have to be executed only on the node on which session is created.

Glusterfind Create

To create a session for a particular instance in the volume, execute the following command:

```
# glusterfind create [-h] [--debug] [--force] <SessionName> <volname> [-
-reset-session-time]
```

where,

--force: is executed when a new node/brick is added to the volume .

--reset-session-time: forces reset of the session time. The next incremental run will start from this time.

--help OR -h: Used to display help for the command.

SessionName: Unique name of a session.

volname: Name of the volume for which the `create` command is executed.

For example:

```
# glusterfind create sess_vol1 vol1
Session sess_vol1 created with volume vol1
```

Glusterfind Pre

To retrieve the list of modified files and directories and store it in the outfile, execute the following command:

```
# glusterfind pre [--debug] [--disable-partial] [--output-prefix
OUTPUT_PREFIX] [--no-encode] [--regenerate-outfile] [--field-separator
FIELD_SEPARATOR] [-h] <session> <volname> <outfile>
```

where,

--disable-partial: Disables the partial-find feature that is enabled by default.

--output-prefix *OUTPUT_PREFIX*: Prefix to the path/name that is specified in the outfile.

--regenerate-outfile: Regenerates a new outfile and discards the outfile generated from the last pre command.

--no-encode: The file paths are encoded by default in the output file. This option disables encoding of file paths.

--field-separator: Specifies the character/s that glusterfind output uses to separate fields. By default this is a single space, but if your file names contain spaces, you may want to change the delimiter so you can parse the output of glusterfind automatically.

--help OR -h: Displays help for the command

session: Unique name of a session.

volname: Name of the volume for which the **`pre`** command is executed.

outfile: Incremental list of modified files.

For example:

```
# glusterfind pre sess_vol1 vol1 /tmp/outfile.txt
Generated output file /tmp/outfile.txt
```

Note

The output format is <TYPE> <PATH1> <PATH2>. Possible type values are, NEW, MODIFY, DELETE and RENAME. PATH2 is applicable only if type is RENAME. For example:

```
NEW file1
NEW dir1%2Ffile2
MODIFY dir3%2Fdir4%2Ftest3
RENAME test1 dir1%2F%2Ftest1new
DELETE test2
```

The example output with **`--no-encode`** option

```
NEW file1
NEW dir1/file2
MODIFY dir3/dir4/test3
RENAME test1 dir1/test1new
DELETE test2
```

Glusterfind Post

The following command is run to update the session time:

```
# glusterfind post [-h] [--debug] <SessionName> <volname>
```

where,

SessionName: Unique name of a session.

volname: Name of the volume for which the **post** command is executed.

For example:

```
# glusterfind post sess_vol1 vol1
Session sess_vol1 with volume vol1 updated
```

Glusterfind List

To list all the active sessions and the corresponding volumes present in the cluster, execute the following command:

```
# glusterfind list [-h] [--session SESSION] [--volume VOLUME] [--debug]
```

where,

--session SESSION: Displays the information related to that session

--volume VOLUME: Displays all the active sessions corresponding to that volume

--help OR -h: Displays help for the command

For example:

```
# glusterfind list
SESSION VOLUME SESSION TIME
--------------------------------------------------
sess_vol1 vol1 2015-06-22 22:22:53
```

Glusterfind Query

The **glusterfind query** subcommand provides a list of changed files based on a specified time stamp. These commands do not check any change log information. Use the **glusterfind query** subcommand when your backup software maintains its own checkpoints and time stamps outside glusterfind.

To retrieve files changed between two timestamps, run the following command:

```
# glusterfind query volname --since-time timestamp1 --end-time timestamp2
output_file.txt
```

Time stamps are expected in seconds since the Linux epoch date (1970-01-01 00:00:00 UTC). Current Linux epoch time can be output by running **echo $(date +'%s')** on the command line.

You can retrieve all files in the volume by running the following command:

```
# glusterfind query volname --full output_file.txt
```

When running a full find operation, you can also retrieve a subset of files according to a tag. For example, to find all new files on a volume, run the following command:

```
# glusterfind query volname --full --tag-for-full-find NEW output_file.txt
```

By default, the output of glusterfind uses a single space to separate fields. If your file names contain spaces, you may want to change the delimiter in order to parse the output of glusterfind automatically. You can set the delimiter to one or more characters by using the **--field-separator** option. The following command sets the field separator to **==**.

```
# gluster query volname --full output_file.txt --field-separator "=="
```

Glusterfind Delete

To clear out all the session information associated with that particular session, execute the following command:

Ensure, no further backups are expected to be taken in a particular session.

```
# glusterfind delete [-h] [--debug] <SessionName> <volname>
```

where,

SessionName: Unique name of a session.

volname: Name of the volume for which the **delete** command is executed.

For example:

```
# glusterfind delete sess_vol1 vol1
Session sess_vol1 with volume vol1 deleted
```

16.1.1. Adding or Replacing a Brick from an Existing Glusterfind Session

When a new brick is added or an existing brick is replaced, execute the **glusterfind create** command with **force** for the existing session to work. For example:

```
# glusterfind create existing-session volname --force
```

Chapter 17. Managing Tiering

Tiering refers to automatic classification and movement of data based on the user I/O access. The tiering feature continuously monitors the workload, identifies hotspots by measuring and analysing the statistics of the activity, and moves frequently accessed data to the highest performance hot tier (such as solid state drives (SSDs)), and inactive data to the lower performing cold tier (such as spinning disks) all without I/O interruption. With tiering, data promotion and automatic rebalancing improve access time for popular files, while demoting infrequently accessed files to the cold tier regulates the hot tier's capacity.

Important

Data is moved, not copied, from one tier to another. When a file is moved to one tier, a copy is not kept on the other tier.

Tiering monitors and identifies the activity level of the data and automatically moves the active and inactive data to the most appropriate storage tier. Moving data between tiers of hot and cold storage is a computationally expensive task. To address this, Red Hat Gluster Storage supports automated promotion and demotion of data within a volume in the background so as to minimize impact on foreground I/O. Data becomes hot or cold based on the rate at which it is accessed. If access to a file increases, it moves to the hot tier or retains its place in the hot tier. If the file is not accessed for a while, it moves to the cold tier, or retains it place in the cold tier. Hence, the data movement can happen in either direction which is based totally on the access frequency.

Different sub-volume types act as hot and cold tiers and data is automatically assigned or reassigned a “temperature” based on the frequency of access. Red Hat Gluster Storage allows attaching fast performing disks as hot tier, uses the existing volume as cold tier, and these hot tier and cold tier forms a single tiered volume. For example, the existing volume may be distributed dispersed on HDDs and the hot tier could be distributed-replicated on SSDs.

Hot Tier

The hot tier is the tiering volume created using better performing subvolumes, an example of which could be SSDs. Frequently accessed data is placed in the highest performance and most expensive hot tier. Hot tier volume could be a distributed volume or distributed-replicated volume.

Warning

Distributed volumes can suffer significant data loss during a disk or server failure because directory contents are spread randomly across the bricks in the volume. Red Hat recommends creating distributed-replicated tier volume.

Cold Tier

The cold tier is the existing Red Hat Gluster Storage volume created using slower storage such as Spinning disks. Inactive or infrequently accessed data is placed in the lowest-cost cold tier.

Data Migration

Tiering automatically migrates files between hot tier and cold tier to improve the storage performance and resource use.

17.1. Tiering Architecture

Tiering provides better I/O performance as a subset of the data is stored in the hot tier. Tiering involves creating a pool of relatively fast/expensive storage devices (example, solid state drives) configured to act as a hot tier, and an existing volume which are relatively slower/cheaper devices configured to act as a cold tier. The tiering translator handles where to place the files and when to migrate files from the cold tier to the hot tier and vice versa.

The following diagrams illustrates how tiering works when attached to a distributed-dispersed volume. Here, the existing distributed-dispersed volume would become a cold-tier and the new fast/expensive storage device would act as a hot tier. Frequently accessed files will be migrated from cold tier to the hot tier for better performance.

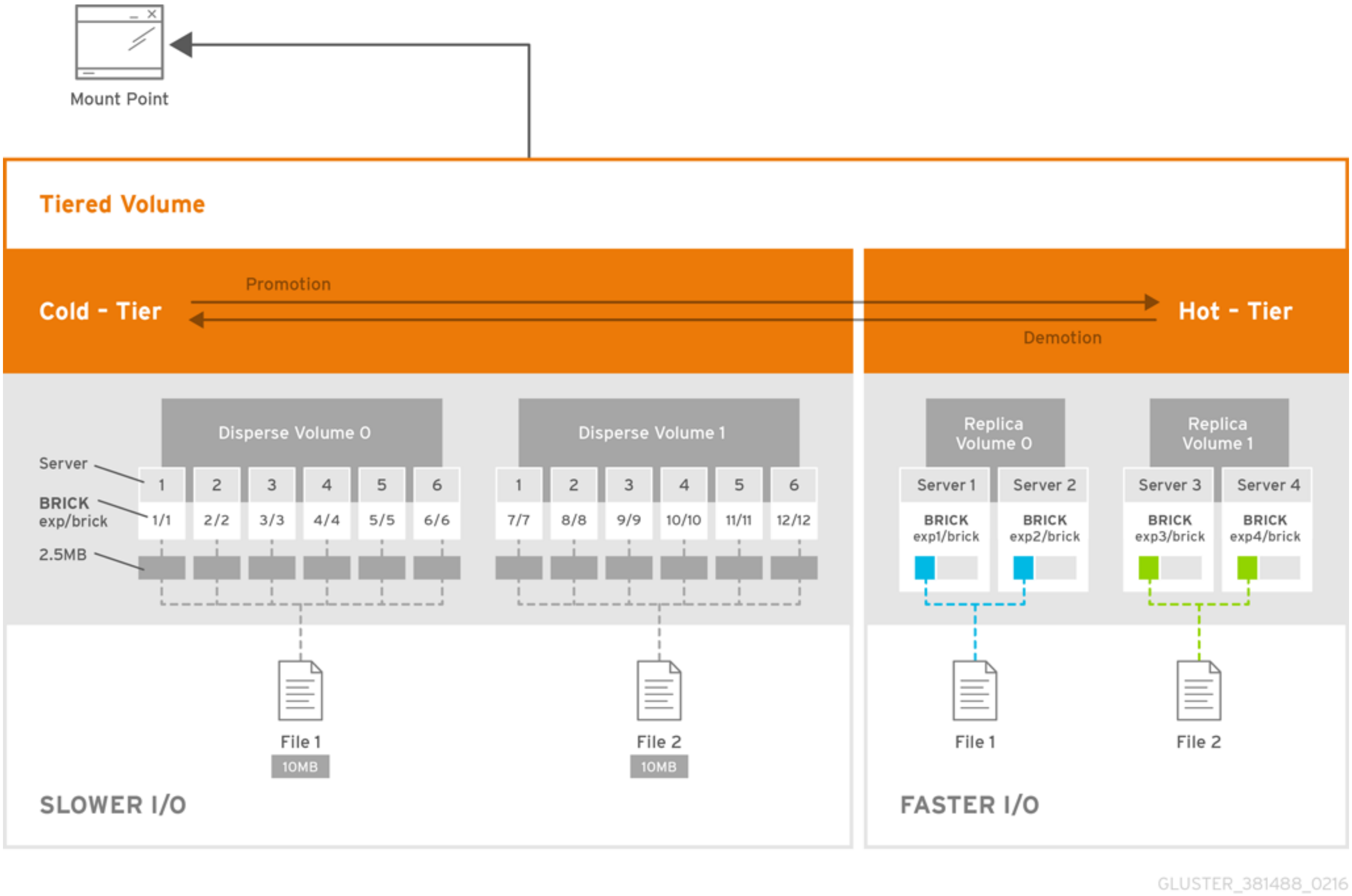


Figure 17.1. Tiering Architecture

17.2. Key Benefits of Tiering

The following are the key benefits of data tiering:

- Automatic classification and movement of files based on the access patterns
- Faster response time and reduced latency
- Better I/O performance
- Improved data-storage efficiency
- Reduced deployment and operating costs

17.3. Tiering Limitations

The following limitations apply to the use Tiering feature:

- Native client support for tiering is limited to Red Hat Enterprise Linux version 6.7, 6.8 and 7.x clients. Tiered volumes cannot be mounted by Red Hat Enterprise Linux 5.x clients.

- Tiering works only with **`cache friendly`** workloads. Attaching a tier volume to a cache unfriendly workload will lead to slow performance. In a **`cache friendly`** workload, most of the reads and writes are accessing a subset of the total amount of data. And, this subset fits on the hot tier. This subset should change only infrequently.

- Tiering feature is supported only on Red Hat Enterprise Linux 7 based Red Hat Gluster Storage. Tiering feature is not supported on Red Hat Enterprise Linux 6 based Red Hat Gluster Storage.

- Only Fuse and gluster-nfs access is supported. Server Message Block (SMB) and nfs-ganesha access to tiered volume is not supported.

- Creating snapshot of a tiered volume is supported. Snapshot clones are not supported with the tiered volumes.

- When you run **`tier detach commit`** or **`tier detach force`**, ongoing I/O operations may fail with a *Transport endpoint is not connected* error.

- Files with hardlinks and softlinks are not migrated.

- Files on which POSIX locks has been taken are not migrated until all locks are released.

- Add brick, remove brick, and rebalance operations are not supported on the tiered volume. For information on expanding a tiered volume, see Section 11.7.1, “Expanding a Tiered Volume” and for information on shrinking a tiered volume, see Section 11.8.2, “Shrinking a Tiered Volume ”

17.4. Attaching a Tier to a Volume

By default, tiering is not enabled on gluster volumes. An existing volume can be modified via a CLI command to have a *hot tier*. You must enable a volume by performing an attach tier operation. The **`attach`** command will declare an existing volume as *cold tier* and creates a new *hot tier* volume which is appended to it. Together, the combination is a single *tiered* volume.

It is highly recommended to provision your storage liberally and generously before attaching a tier. You create a normal volume and then *attach* bricks to it, which are the *hot tier*:

1. Attach the tier to the volume by executing the following command:

   ```
   # gluster volume tier VOLNAME attach [replica COUNT] NEW-BRICK...
   ```

   For example,

   ```
   # gluster volume tier test-volume attach replica 2
   server1:/rhgs/brick5/b1 server2:/rhgs/brick6/b2
   server1:/rhgs/brick7/b3 server2:/rhgs/brick8/b4
   ```

2. Run **`gluster volume info`** command to optionally display the volume information.

   The command output displays information similar to the following:

```
# gluster volume info test-volume
Volume Name: test-volume
Type: Tier
Status: Started
Number of Bricks: 8
Transport-type: tcp
Hot Tier :
Hot Tier Type : Distributed-Replicate
Number of Bricks: 2 x 2 = 4
Brick1: server1:/rhgs/brick5/b1
Brick2: server2:/rhgs/brick6/b2
Brick3: server1:/rhgs/brick7/b3
Brick4: server2:/rhgs/brick8/b4
Cold Tier:
Cold Tier Type : Distributed-Replicate
Number of Bricks: 2 x 2 = 4
Brick5: server1:/rhgs/brick1/b5
Brick6: server2:/rhgs/brick2/b6
Brick7: server1:/rhgs/brick3/b7
Brick8: server2:/rhgs/brick4/b8
Options Reconfigured:
cluster.watermark-low: 70
cluster.watermark-hi: 90
cluster.tier-demote-frequency: 45
cluster.tier-mode: cache
features.ctr-enabled: on
performance.readdir-ahead: on
```

The tier start command is triggered automatically after the tier has been attached. In some cases, if the tier process has not started you must start it manually using the **gluster volume tier *VOLNAME* start force** command.

17.4.1. Attaching a Tier to a Geo-replicated Volume

You can attach a tier volume to the master volume of the geo-replication session for better performance.

Important

A crash has been observed in the Slave mounts when **performance.quick-read** option is enabled and geo-replicated from a tiered master volume. If the master volume is a tiered volume, you must disable the **performance.quick-read** option in the Slave Volume using the following command:

```
# gluster volume set Slavevol performance.quick-read off
```

1. Stop geo-replication between the master and slave, using the following command:

   # gluster volume geo-replication *MASTER_VOL SLAVE_HOST::SLAVE_VOL* stop

   For example:

   ```
   # gluster volume geo-replication Volume1 example.com::slave-vol stop
   ```

2. Attach the tier to the volume using the following command:

   ```
   # gluster volume tier VOLNAME attach [replica COUNT] NEW-BRICK...
   ```

   For example, to create a distributed-replicated tier volume with replica count two:

   ```
   # gluster volume tier test-volume attach replica 2
   server1:/rhgs/brick1/b1 server2:/rhgs/brick2/b2
   server1:/rhgs/brick3/b3 server2:/rhgs/brick4/b4
   ```

3. Restart the geo-replication sessions, using the following command:

   ```
   # gluster volume geo-replication MASTER_VOL SLAVE_HOST::SLAVE_VOL start
   ```

   For example

   ```
   # gluster volume geo-replication Volume1 example.com::slave-vol start
   ```

4. Verify whether geo-replication session has started with tier's bricks, using the following command:

   ```
   # gluster volume geo-replication MASTER_VOL SLAVE_HOST::SLAVE_VOL status
   ```

   For example,

   ```
   # gluster volume geo-replication Volume1 example.com::slave-vol status
   ```

17.5. Configuring a Tiering Volume

Tiering volume has several configuration options. You may set tier volume configuration options with the following usage:

```
# gluster volume set VOLNAME key value
```

17.5.1. Configuring Watermarks

When the tier volume is configured to use the **cache** mode, the configured watermark values and the percentage of the hot tier that is full determine whether a file will be promoted or demoted. The **cluster.watermark-low** and **cluster.watermark-hi** volume options set the lower and upper watermark values respectively for a tier volume.

The promotion and demotion of files is determined by how full the hot tier is. Data accumulates on the hot tier until it reaches the low watermark, even if it is not accessed for a period of time. This prevents files from being demoted unnecessarily when there is plenty on free space on the hot tier. When the hot tier is fuller than the lower watermark but less than the high watermark, data is randomly promoted and demoted where the likelihood of promotion decreases as the tier becomes fuller; the opposite holds for demotion. If the hot tier is fuller than the high watermark, promotions stop and demotions happen more frequently in order to free up space.

The following diagram illustrates how cache mode works and the example values you can set.

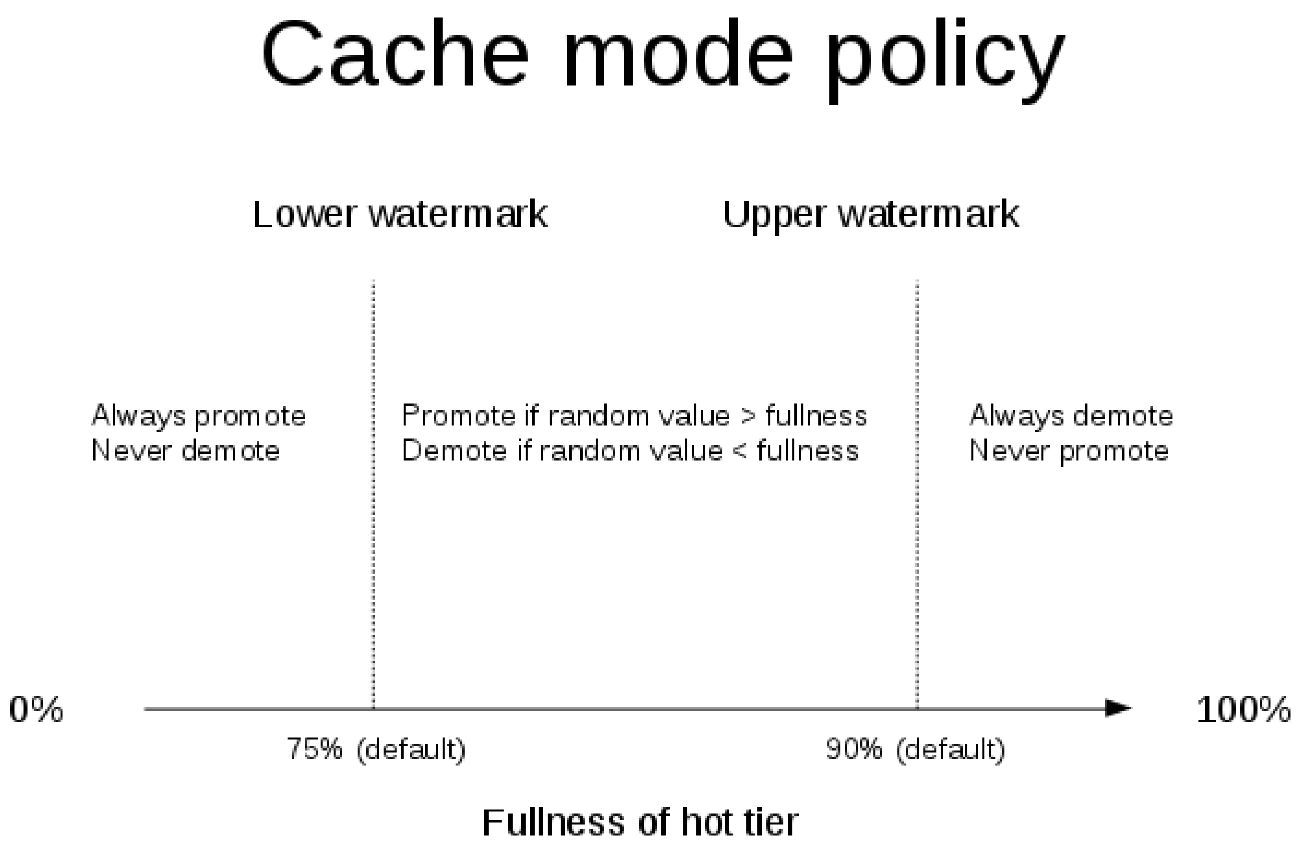


Figure 17.2. Tiering Watermarks

To set the percentage for promotion and demotion of files, run the following commands:

```
# gluster volume set VOLNAME cluster.watermark-hi value
```

```
# gluster volume set VOLNAME cluster.watermark-low value
```

17.5.2. Configuring Promote and Demote Frequency

You can configure how frequently the files are to be checked for promotion and demotion of files. The check is based on whether the file was accessed or not in the last *n* seconds. If the promote/demote frequency is not set, then the default value for promote frequency is 120 seconds and demote frequency is 3600 seconds.

To set the frequency for the promotion and demotion of files, run the following command:

```
# gluster volume set VOLNAME cluster.tier-demote-frequency value_in_seconds
```

```
# gluster volume set VOLNAME cluster.tier-promote-frequency value_in_seconds
```

17.5.3. Configuring Read and Write Frequency

You can configure the number of reads and writes in a promotion/demotion cycle, that would mark a file **HOT** for promotion. Any file that has read or write hits less than this value will be considered as **COLD** and will be demoted. If the read/write access count is not set, then the default count is set to 0.

Set the read and write frequency threshold by executing the following command:

```
# gluster volume set VOLNAME cluster.write-freq-threshold value
```

Note

The value of 0 indicates that the threshold value is not considered. Any value in the range of 1-1000 denotes the number of times the contents of file must be modified to consider for promotion or demotion...

```
# gluster volume set VOLNAME cluster.read-freq-threshold value
```

Note

The value of 0 indicates that the threshold value is not considered. Any value in the range of 1-1000 denotes the number of times the contents of file contents have been accessed to consider for promotion or demotion.

17.5.4. Configuring Target Data Size

The maximum amount of data that may be migrated in any direction in one promotion/demotion cycle from each node can be configured using the following command:

```
# gluster volume set VOLNAME cluster.tier-max-mb value_in_mb
```

If the **cluster.tier-max-mb** count is not set, then the default data size is set to 4000 MB.

17.5.5. Configuring the File Count per Cycle

The maximum number of files that may be migrated in any direction in one promotion/demotion cycle from each node can be configured using the following command:

```
 # gluster volume set VOLNAME cluster.tier-max-files count
```

If the **cluster.tier-max-files** count is not set, then the default count is set to 10000.

17.6. Displaying Tiering Status Information

The status command displays the tiering volume information.

```
# gluster volume tier VOLNAME status
```

For example,

```
# gluster volume tier test-volume status
Node                 Promoted files       Demoted files        Status
---------            ---------            ---------            ---------
localhost            1                    5                    in progress
server1              0                    2                    in
progress
Tiering Migration Functionality: test-volume: success
```

17.7. Detaching a Tier from a Volume

To detach a tier, perform the following steps:

1. Start the detach tier by executing the following command:

   # gluster volume tier *VOLNAME* detach start

   For example,

   ```
   # gluster volume tier test-volume detach start
   ```

2. Monitor the status of detach tier until the status displays the status as complete.

   # gluster volume tier *VOLNAME* detach status

   For example,

   ```
   # gluster volume tier test-volume detach status
   Node Rebalanced-files          size       scanned      failures
   skipped               status       run time in secs
   --------      -----------   -----------   -----------   -----------
   -----------         ------------     --------------
   localhost           0        0Bytes             0             0
   0            completed               0.00
   server1             0        0Bytes             0             0
   0            completed               1.00
   server2             0        0Bytes             0             0
   0            completed               0.00
   server1             0        0Bytes             0             0
   0            completed
   server2             0        0Bytes             0             0
   0            completed
   ```

   > **Note**
   >
   > It is possible that some files are not migrated to the cold tier on a detach operation for various reasons like POSIX locks being held on them. Check for files on the hot tier bricks and you can either manually move the files, or turn off applications (which would presumably unlock the files) and stop/start detach tier, to retry.

3. When the tier is detached successfully as shown in the previous status command, run the following command to commit the tier detach:

   # gluster volume tier *VOLNAME* detach commit

   For example,

   ```
   # gluster volume tier test-volume detach commit
   Removing tier can result in data loss. Do you want to Continue? (y/n)
   y
   volume detach-tier commit: success
   Check the detached 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.
   ```

Note

When you run **`tier detach commit`** or **`tier detach force`**, ongoing I/O operations may fail with a *Transport endpoint is not connected* error.

After the detach tier commit is completed, you can verify that the volume is no longer a tier volume by running **`gluster volume info`** command.

17.7.1. Detaching a Tier of a Geo-replicated Volume

1. Start the detach tier by executing the following command:

   `# gluster volume tier` ***`VOLNAME`*** **`detach start`**

   For example,

   ```
   # gluster volume tier test-volume detach start
   ```

2. Monitor the status of detach tier until the status displays the status as complete.

   `# gluster volume tier` ***`VOLNAME`*** **`detach status`**

   For example,

   ```
   # gluster volume tier test-volume detach status
   Node Rebalanced-files          size       scanned      failures
   skipped               status       run time in secs
   --------      -----------   -----------   -----------   -----------
   -----------         ------------     --------------
   localhost           0        0Bytes             0             0
   0            completed               0.00
   server1             0        0Bytes             0             0
   0            completed               1.00
   server2             0        0Bytes             0             0
   0            completed               0.00
   server1             0        0Bytes             0             0
   0            completed
   server2             0        0Bytes             0             0
   0            completed
   ```

   Note

   There could be some number of files that were not moved. Such files may have been locked by the user, and that prevented them from moving to the cold tier on the detach operation. You must check for such files. If you find any such files, you can either manually move the files, or turn off applications (which would presumably unlock the files) and stop/start detach tier, to retry.

3. Set a checkpoint on a geo-replication session to ensure that all the data in that cold-tier is synced to the slave. For more information on geo-replication checkpoints, see Section 10.4.4.1, “Geo-replication Checkpoints”.

# gluster volume geo-replication *MASTER_VOL SLAVE_HOST::SLAVE_VOL* config checkpoint now

For example,

```
# gluster volume geo-replication Volume1 example.com::slave-vol config
checkpoint now
```

4. Use the following command to verify the checkpoint completion for the geo-replication session

   # gluster volume geo-replication *MASTER_VOL SLAVE_HOST::SLAVE_VOL* status detail

5. Stop geo-replication between the master and slave, using the following command:

   # gluster volume geo-replication *MASTER_VOL SLAVE_HOST::SLAVE_VOL* stop

   For example:

   ```
   # gluster volume geo-replication Volume1 example.com::slave-vol stop
   ```

6. Commit the detach tier operation using the following command:

   # gluster volume tier *VOLNAME* detach commit

   For example,

   ```
   # gluster volume tier test-volume detach commit
   Removing tier can result in data loss. Do you want to Continue? (y/n)
   y
   volume detach-tier commit: success
   Check the detached 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.
   ```

   After the detach tier commit is completed, you can verify that the volume is no longer a tier volume by running **gluster volume info** command.

7. Restart the geo-replication sessions, using the following command:

   # gluster volume geo-replication *MASTER_VOL SLAVE_HOST::SLAVE_VOL* start

   For example,

   ```
   # gluster volume geo-replication Volume1 example.com::slave-vol start
   ```

Part V. Monitor and Tune

Chapter 18. Monitoring Red Hat Gluster Storage

Red Hat Gluster Storage 3.4 delivers Red Hat Gluster Storage Web Administration. This intuitive GUI delivers monitoring and metrics capabilities with data that is derived from the Grafana platform. Red Hat Gluster Storage Web Administration allows you to view and customize metrics information related to cluster health, status, and performance. Event and alert notification through email is also available. Red Hat recommends using Gluster Storage Web Administration to monitor your Red Hat Gluster Storage environment. The existing Red Hat Gluster Storage Console management infrastructure is supported through the current Red Hat Gluster Storage 3.x life cycle, which ends on October 31, 2019.

To view metrics data and monitor Red Hat Gluster Storage servers with Red Hat Gluster Storage Web Administration, see the following publications:

- *Red Hat Gluster Storage Web Administration Monitoring Guide*
- *Red Hat Gluster Storage Web Administration Quick Start Guide*

Warning

As of Red Hat Gluster Storage 3.4, Nagios is considered deprecated. Nagios plugins and Nagios server are no longer maintained and would not be provided in releases post Red Hat Gluster Storage 3.4. Nagios remains supported for this release, but Red Hat no longer recommends its use, and plans to remove support in future versions of Red Hat Gluster Storage.

Nagios is being deprecated because of the limited capabilities of monitoring and aggregation of results for a gluster cluster. These limitations are addressed in Red Hat Gluster Storage Web Administration.

Red Hat Gluster Storage users need to set up Red Hat Gluster Storage Web Administration in order to monitor a cluster. There is no migration path for the data collected in Nagios.

Information about Red Hat Gluster Storage Web Administration is available in the following publications:

- *Red Hat Gluster Storage Web Administration Monitoring Guide*
- *Red Hat Gluster Storage Web Administration Quick Start Guide*

You can also monitor Red Hat Gluster Storage servers on Nagios platform to monitor Red Hat Gluster Storage trusted storage pool, hosts, volumes, and services. You can monitor utilization, status, alerts and notifications for status and utilization changes.

For more information on Nagios software, refer *Nagios Documentation*.

Using Nagios, the physical resources, logical resources, and processes (CPU, Memory, Disk, Network, Swap, cluster, volume, brick, Host, Volumes, Brick, nfs, shd, quotad, ctdb, smb, glusterd, quota, geo-replication, self-heal,and server quorum) can be monitored. You can view the utilization and status through Nagios Server GUI.

Red Hat Gluster Storage trusted storage pool monitoring can be setup in one of the three deployment scenarios listed below:

- Nagios deployed on Red Hat Gluster Storage node.
- Nagios deployed on Red Hat Gluster Storage Console node.
- Nagios deployed on Red Hat Enterprise Linux node.

This chapter describes the procedures for deploying Nagios on Red Hat Gluster Storage node and Red Hat Enterprise Linux node. For information on deploying Nagios on Red Hat Gluster Storage Console node, see *Red Hat Gluster Storage Console Administration Guide*.

The following diagram illustrates deployment of Nagios on Red Hat Gluster Storage node.

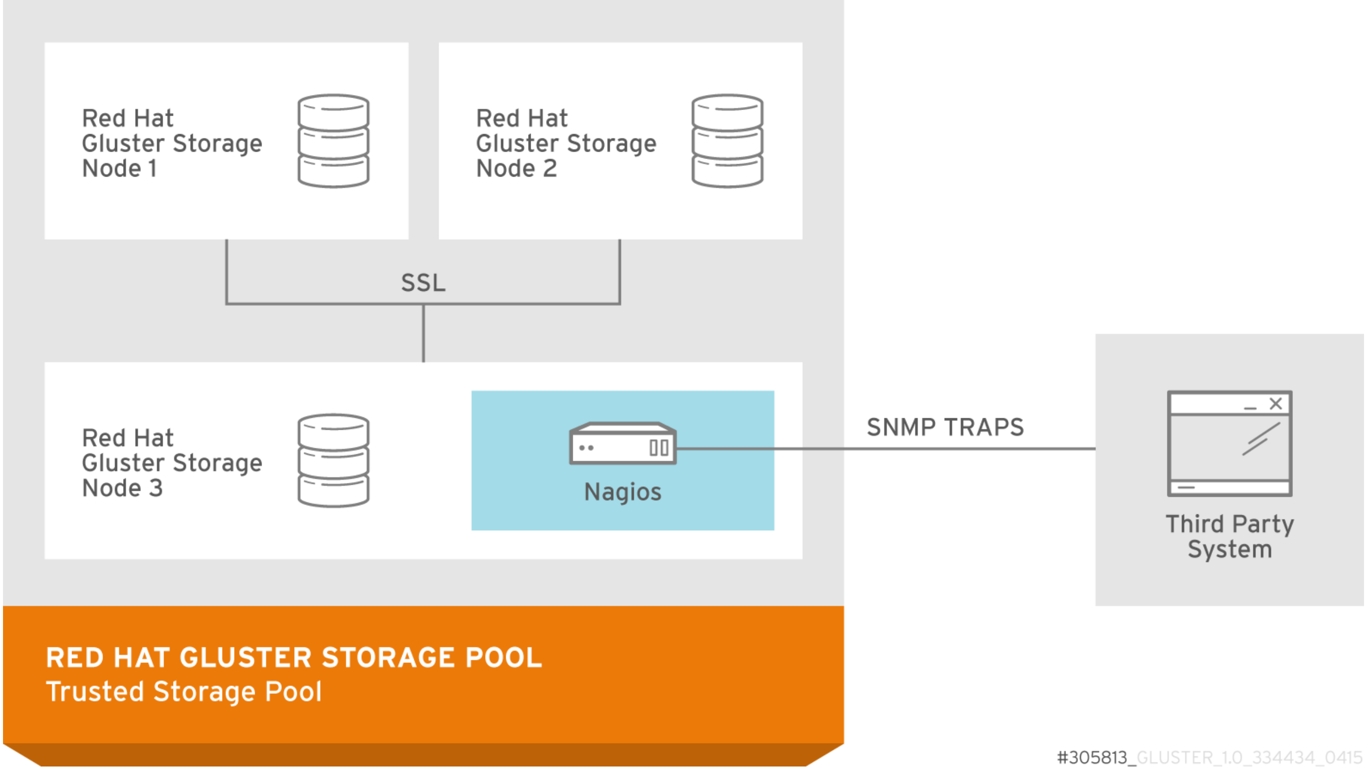

Red Hat Gluster Storage Node 1
Red Hat Gluster Storage Node 2
SSL
Red Hat Gluster Storage Node 3
Nagios
SNMP TRAPS
Third Party System
RED HAT GLUSTER STORAGE POOL
Trusted Storage Pool
#305813_GLUSTER_1.0_334434_0415

Figure 18.1. Nagios deployed on Red Hat Gluster Storage node

The following diagram illustrates deployment of Nagios on Red Hat Enterprise Linux node.

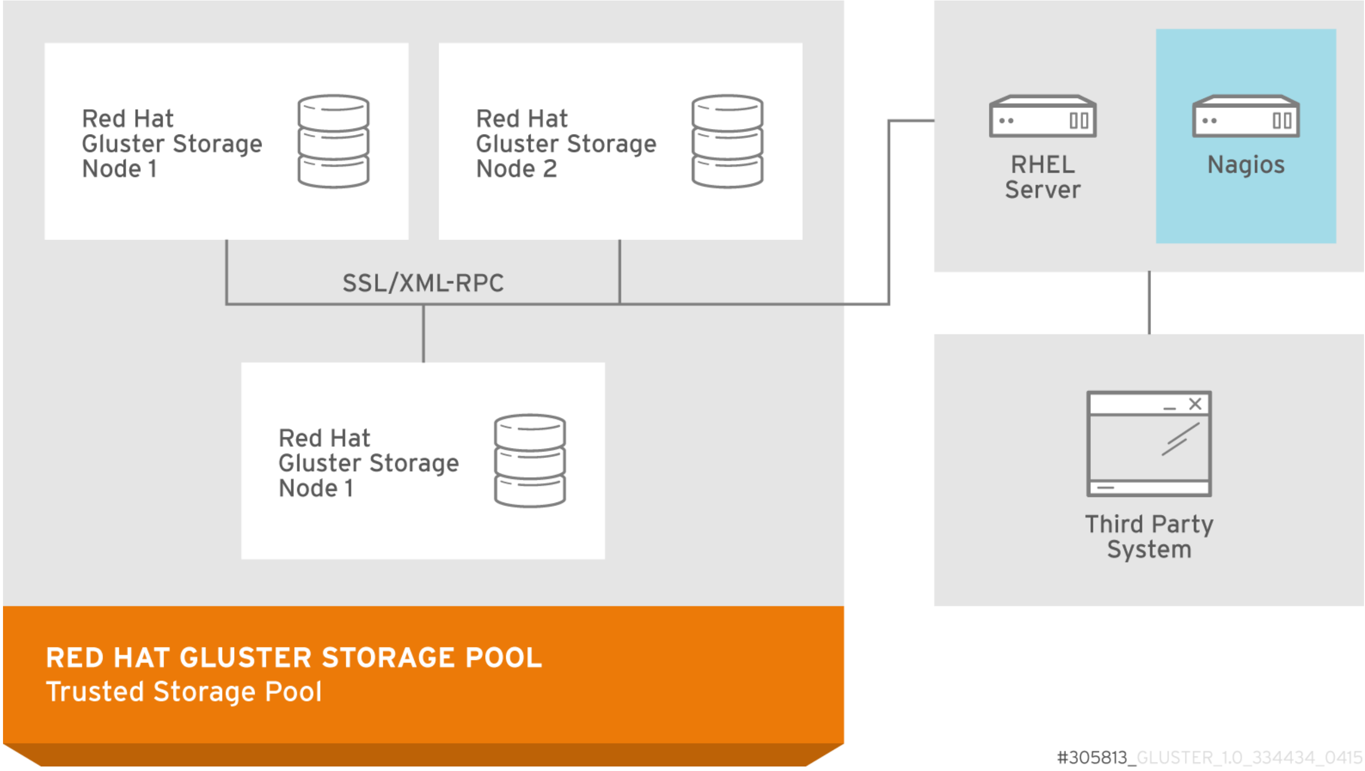

Red Hat Gluster Storage Node 1
Red Hat Gluster Storage Node 2
SSL/XML-RPC
Red Hat Gluster Storage Node 1
RHEL Server
Nagios
Third Party System
RED HAT GLUSTER STORAGE POOL
Trusted Storage Pool
#305813_GLUSTER_1.0_334434_0415

Figure 18.2. Nagios deployed on Red Hat Enterprise Linux node

18.1. Prerequisites

Ensure that you register using Subscription Manager or Red Hat Network Classic (RHN) and enable the Nagios repositories before installing the Nagios Server.

> **Note**
>
> Register using Red Hat Network (RHN) Classic only if you are a Red Hat Satellite user.

- Registering using Subscription Manager and enabling Nagios repositories
  - To install Nagios on Red Hat Gluster Storage node, subscribe to **`rhs-nagios-3-for-rhel-6-server-rpms`** repository.
  - To install Nagios on Red Hat Enterprise Linux node, subscribe to **`rhel-6-server-rpms`**, **`rhs-nagios-3-for-rhel-6-server-rpms`** repositories.
  - To install Nagios on Red Hat Gluster Storage node based on RHEL7, subscribe to **`rh-gluster-3-nagios-for-rhel-7-server-rpms`** repository.
  - To install Nagios on Red Hat Enterprise Linux node, subscribe to **`rhel-7-server-rpms`**, **`rh-gluster-3-nagios-for-rhel-7-server-rpms`** repositories.
- Registering using Red Hat Network (RHN) Classic and subscribing to Nagios channels
  - To install Nagios on Red Hat Gluster Storage node, subscribe to **`rhel-x86_64-server-6-rhs-nagios-3`** channel.
  - To install Nagios on Red Hat Gluster Storage node, subscribe to **`rhel-x86_64-server-7-rh-gluster-3-nagios`** channel.
  - To install Nagios on Red Hat Enterprise Linux node, subscribe to **`rhel-x86_64-server-6`**, **`rhel-x86_64-server-6-rhs-nagios-3`** channels.
  - To install Nagios on Red Hat Enterprise Linux node, subscribe to **`rhel-x86_64-server-7`**, **`rhel-x86_64-server-7-rh-gluster-3-nagios`** channels.

> **Note**
>
> Once nagios is installed on Red Hat Gluster Storage or RHEL node, verify that the following booleans are ON by running the **`getsebool -a | grep nagios`** command:
>
> - **`nagios_run_sudo --> on`**
> - **`nagios_run_pnp4nagios --> on`**

18.2. Installing Nagios

The Nagios monitoring system is used to provide monitoring and alerts for the Red Hat Gluster Storage network and infrastructure. Installing Nagios installs the following components.

nagios

Core program, web interface and configuration files for Nagios server.

python-cpopen

Python package for creating sub-process in simple and safe manner.

python-argparse

Command line parser for python.

libmcrypt

Encryptions algorithm library.

rrdtool

Round Robin Database Tool to store and display time-series data.

pynag

Python modules and utilities for Nagios plugins and configuration.

check-mk

General purpose Nagios-plugin for retrieving data.

mod_python

An embedded Python interpreter for the Apache HTTP Server.

nrpe

Monitoring agent for Nagios.

nsca

Nagios service check acceptor.

nagios-plugins

Common monitoring plug-ins for nagios.

gluster-nagios-common

Common libraries, tools, configurations for Gluster node and Nagios server add-ons.

nagios-server-addons

Gluster node management add-ons for Nagios.

18.2.1. Installing Nagios Server

Use the following command to install Nagios server:

```
# yum install nagios-server-addons
```

You must install Nagios on the node which would be used as the Nagios server.

18.2.2. Configuring Red Hat Gluster Storage Nodes for Nagios

Configure all the Red Hat Gluster Storage nodes, including the node on which the Nagios server is installed.

Note

If SELinux is configured, the sebooleans must be enabled on all Red Hat Gluster Storage nodes and the node on which Nagios server is installed.

Enable the following sebooleans on Red Hat Enterprise Linux node if Nagios server is installed.

```
# setsebool -P logging_syslogd_run_nagios_plugins on
# setsebool -P nagios_run_sudo on
```

To configure the nodes, follow the steps given below:

1. In **/etc/nagios/nrpe.cfg** file, add the central Nagios server IP address as shown below:

   ```
   allowed_hosts=127.0.0.1, NagiosServer-HostName-or-IPaddress
   ```

2. Restart the **NRPE** service using the following command:

   ```
   # service nrpe restart
   ```

   Note

   - The host name of the node is used while configuring Nagios server using auto-discovery. To view the host name, run **hostname** command.
   - Ensure that the host names are unique.

3. Start the **glusterpmd** service using the following command:

   ```
   # service glusterpmd start
   ```

   To start **glusterpmd** service automatically when the system reboots, run **chkconfig --add glusterpmd** command.

   You can start the **glusterpmd** service using **service glusterpmd start** command and stop the service using **service glusterpmd stop** command.

   The **glusterpmd** service is a Red Hat Gluster Storage process monitoring service running in every Red Hat Gluster Storage node to monitor glusterd, self heal, smb, quotad, ctdbd and brick services and to alert the user when the services go down. The **glusterpmd** service sends its managing services detailed status to the Nagios server whenever there is a state change on any of its managing services.

   This service uses **/etc/nagios/nagios_server.conf** file to get the Nagios server name and the local host name given in the Nagios server. The **nagios_server.conf** is configured by auto-discovery.

18.3. Monitoring Red Hat Gluster Storage Trusted Storage Pool

This section describes how you can monitor Gluster storage trusted pool.

18.3.1. Configuring Nagios

Auto-Discovery is a python script which automatically discovers all the nodes and volumes in the cluster. It also creates Nagios configuration to monitor them. By default, it runs once in 24 hours to synchronize the Nagios configuration from Red Hat Gluster Storage Trusted Storage Pool configuration.

For more information on Nagios Configuration files, see Chapter 22, *Nagios Configuration Files*

> **Note**
>
> Before configuring Nagios using **`configure-gluster-nagios`** command, ensure that all the Red Hat Gluster Storage nodes are configured as mentioned in Section 18.2.2, “Configuring Red Hat Gluster Storage Nodes for Nagios”.

1. Execute the **`configure-gluster-nagios`** command manually on the Nagios server using the following command:

   ```
   # configure-gluster-nagios -c cluster-name -H HostName-or-IP-address
   ```

   For ***`-c`***, provide a cluster name (a logical name for the cluster) and for ***`-H`***, provide the host name or ip address of a node in the Red Hat Gluster Storage trusted storage pool.

2. Perform the steps given below when **`configure-gluster-nagios`** command runs:

   a. Confirm the configuration when prompted.

   b. Enter the current Nagios server host name or IP address to be configured all the nodes.

   c. Confirm restarting Nagios server when prompted.

      ```
      # configure-gluster-nagios -c demo-cluster -H HostName-or-IP-
      address
      Cluster configurations changed
      Changes :
      Hostgroup demo-cluster - ADD
      Host demo-cluster - ADD
        Service - Volume Utilization - vol-1 -ADD
        Service - Volume Split-Brain - vol-1 -ADD
        Service - Volume Status - vol-1 -ADD
        Service - Volume Utilization - vol-2 -ADD
        Service - Volume Status - vol-2 -ADD
        Service - Cluster Utilization -ADD
        Service - Cluster - Quorum -ADD
        Service - Cluster Auto Config -ADD
      Host Host_Name - ADD
        Service - Brick Utilization - /bricks/vol-1-5 -ADD
        Service - Brick - /bricks/vol-1-5 -ADD
        Service - Brick Utilization - /bricks/vol-1-6 -ADD
        Service - Brick - /bricks/vol-1-6 -ADD
        Service - Brick Utilization - /bricks/vol-2-3 -ADD
        Service - Brick - /bricks/vol-2-3 -ADD
      Are you sure, you want to commit the changes? (Yes, No) [Yes]:
      ```

```
Enter Nagios server address [Nagios_Server_Address]:
Cluster configurations synced successfully from host ip-address
Do you want to restart Nagios to start monitoring newly
discovered entities? (Yes, No) [Yes]:
Nagios re-started successfully
```

All the hosts, volumes and bricks are added and displayed.

3. Login to the Nagios server GUI using the following URL.

```
https://NagiosServer-HostName-or-IPaddress/nagios
```

Note

- The default Nagios user name and password is *nagiosadmin / nagiosadmin*.
- You can manually update/discover the services by executing the **`configure-gluster-nagios`** command or by running **`Cluster Auto Config`** service through Nagios Server GUI.
- If the node with which auto-discovery was performed is down or removed from the cluster, run the **`configure-gluster-nagios`** command with a different node address to continue discovering or monitoring the nodes and services.
- If new nodes or services are added, removed, or if snapshot restore was performed on Red Hat Gluster Storage node, run **`configure-gluster-nagios`** command.

18.3.2. Verifying the Configuration

1. Verify the updated configurations using the following command:

```
# nagios -v /etc/nagios/nagios.cfg
```

If error occurs, verify the parameters set in **`/etc/nagios/nagios.cfg`** and update the configuration files.

2. Restart Nagios server using the following command:

```
# service nagios restart
```

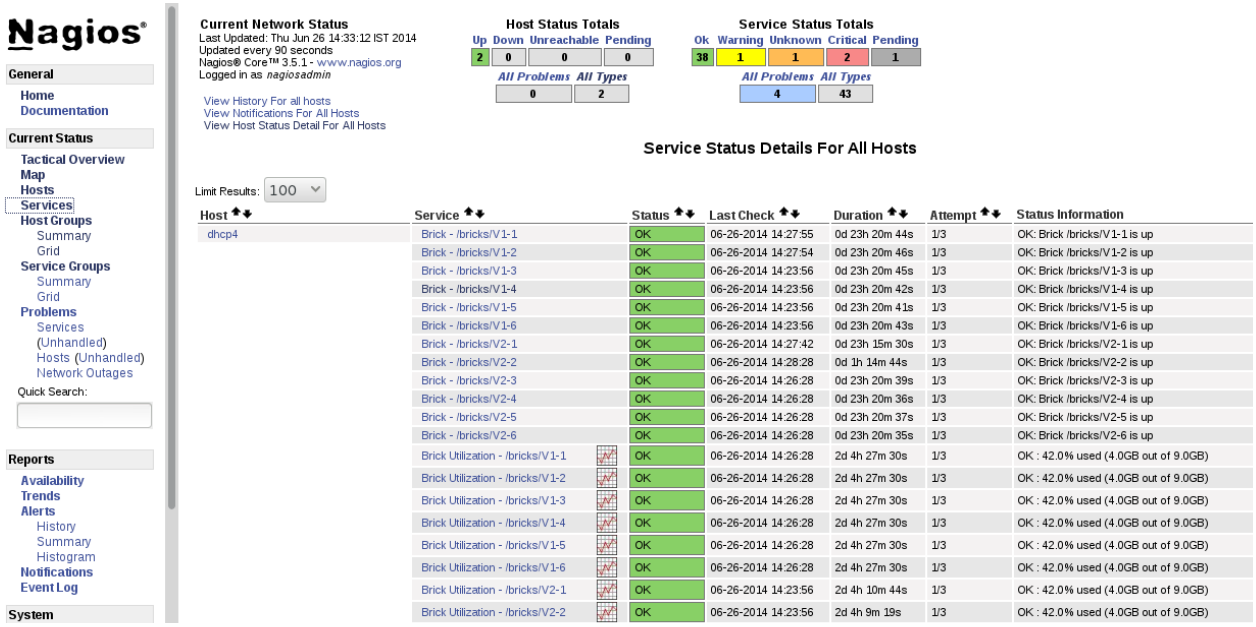
3. Log into the Nagios server GUI using the following URL with the Nagios Administrator user name and password.

```
https://NagiosServer-HostName-or-IPaddress/nagios
```

Note

To change the default password, see *Changing Nagios Password* section in *Red Hat Gluster Storage Administration Guide.*

4. Click **`Services`** in the left pane of the Nagios server GUI and verify the list of hosts and services displayed.



Current Network Status
Last Updated: Thu Jun 26 14:33:12 IST 2014
Updated every 90 seconds
Nagios® Core™ 3.5.1 - www.nagios.org
Logged in as *nagiosadmin*

Host Status Totals

Up	Down	Unreachable	Pending
2	0	0	0

All Problems	*All Types*
0	2

Service Status Totals

Ok	Warning	Unknown	Critical	Pending
38	1	1	2	1

All Problems	*All Types*
4	43

Service Status Details For All Hosts

Host	Service	Status	Last Check	Duration	Attempt	Status Information
dhcp4	Brick - /bricks/V1-1	OK	06-26-2014 14:27:55	0d 23h 20m 44s	1/3	OK: Brick /bricks/V1-1 is up
	Brick - /bricks/V1-2	OK	06-26-2014 14:27:54	0d 23h 20m 46s	1/3	OK: Brick /bricks/V1-2 is up
	Brick - /bricks/V1-3	OK	06-26-2014 14:23:56	0d 23h 20m 45s	1/3	OK: Brick /bricks/V1-3 is up
	Brick - /bricks/V1-4	OK	06-26-2014 14:23:56	0d 23h 20m 42s	1/3	OK: Brick /bricks/V1-4 is up
	Brick - /bricks/V1-5	OK	06-26-2014 14:23:56	0d 23h 20m 41s	1/3	OK: Brick /bricks/V1-5 is up
	Brick - /bricks/V1-6	OK	06-26-2014 14:23:56	0d 23h 20m 43s	1/3	OK: Brick /bricks/V1-6 is up
	Brick - /bricks/V2-1	OK	06-26-2014 14:27:42	0d 23h 15m 30s	1/3	OK: Brick /bricks/V2-1 is up
	Brick - /bricks/V2-2	OK	06-26-2014 14:28:28	0d 1h 14m 44s	1/3	OK: Brick /bricks/V2-2 is up
	Brick - /bricks/V2-3	OK	06-26-2014 14:26:28	0d 23h 20m 39s	1/3	OK: Brick /bricks/V2-3 is up
	Brick - /bricks/V2-4	OK	06-26-2014 14:26:28	0d 23h 20m 36s	1/3	OK: Brick /bricks/V2-4 is up
	Brick - /bricks/V2-5	OK	06-26-2014 14:26:28	0d 23h 20m 37s	1/3	OK: Brick /bricks/V2-5 is up
	Brick - /bricks/V2-6	OK	06-26-2014 14:26:28	0d 23h 20m 35s	1/3	OK: Brick /bricks/V2-6 is up
	Brick Utilization - /bricks/V1-1	OK	06-26-2014 14:26:28	2d 4h 27m 30s	1/3	OK : 42.0% used (4.0GB out of 9.0GB)
	Brick Utilization - /bricks/V1-2	OK	06-26-2014 14:26:28	2d 4h 27m 30s	1/3	OK : 42.0% used (4.0GB out of 9.0GB)
	Brick Utilization - /bricks/V1-3	OK	06-26-2014 14:26:28	2d 4h 27m 30s	1/3	OK : 42.0% used (4.0GB out of 9.0GB)
	Brick Utilization - /bricks/V1-4	OK	06-26-2014 14:26:28	2d 4h 27m 30s	1/3	OK : 42.0% used (4.0GB out of 9.0GB)
	Brick Utilization - /bricks/V1-5	OK	06-26-2014 14:26:28	2d 4h 27m 30s	1/3	OK : 42.0% used (4.0GB out of 9.0GB)
	Brick Utilization - /bricks/V1-6	OK	06-26-2014 14:26:28	2d 4h 27m 30s	1/3	OK : 42.0% used (4.0GB out of 9.0GB)
	Brick Utilization - /bricks/V2-1	OK	06-26-2014 14:23:56	2d 4h 10m 44s	1/3	OK : 42.0% used (4.0GB out of 9.0GB)
	Brick Utilization - /bricks/V2-2	OK	06-26-2014 14:23:56	2d 4h 9m 19s	1/3	OK : 42.0% used (4.0GB out of 9.0GB)

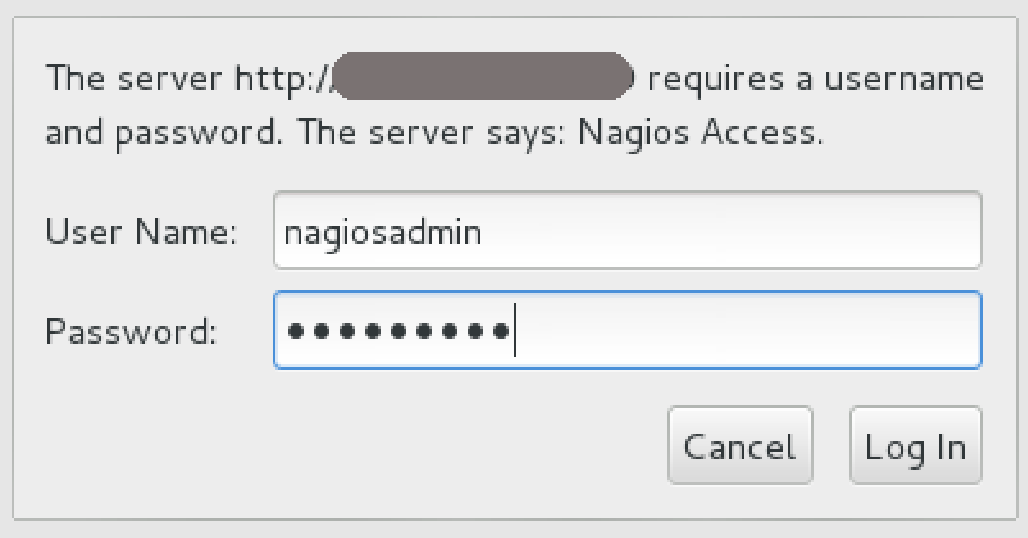
Figure 18.3. Nagios Services

18.3.3. Using Nagios Server GUI

You can monitor Red Hat Gluster Storage trusted storage pool through Nagios Server GUI.

To view the details, log into the Nagios Server GUI by using the following URL.

```
https://NagiosServer-HostName-or-IPaddress/nagios
```



The server http:/[redacted] requires a username and password. The server says: Nagios Access.

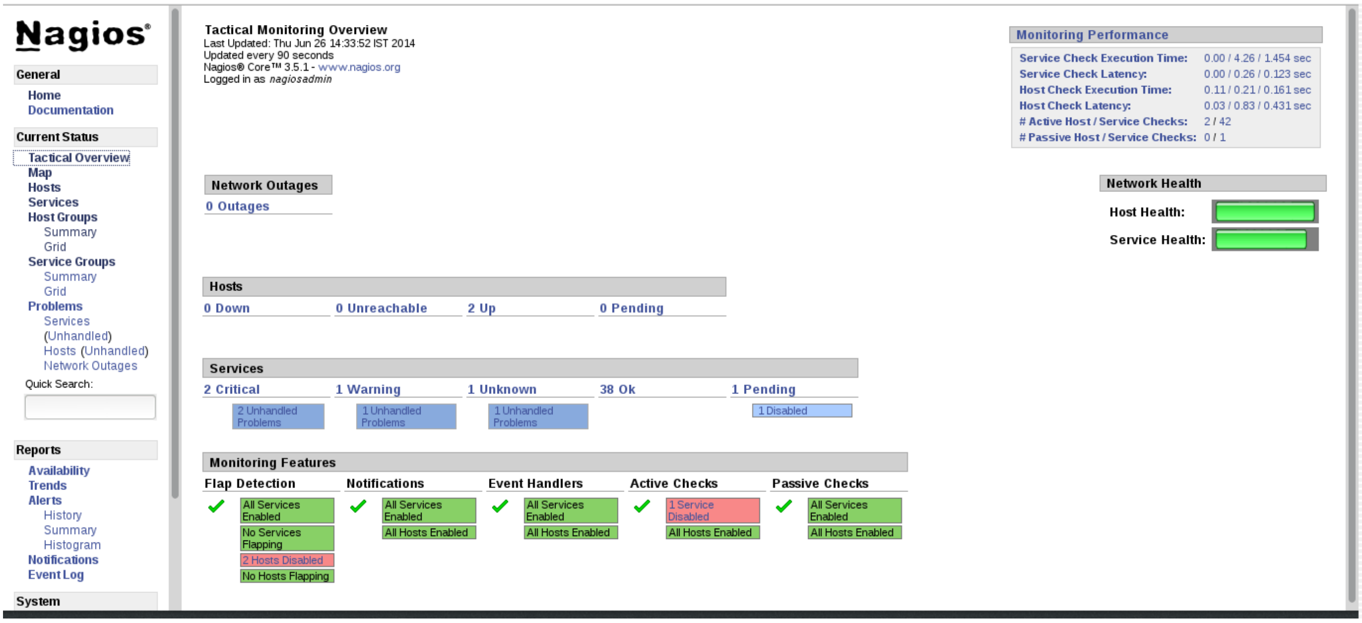
User Name: nagiosadmin

Password: ●●●●●●●●●●

Figure 18.4. Nagios Login

Cluster Overview

To view the overview of the hosts and services being monitored, click **`Tactical Overview`** in the left pane. The overview of Network Outages, Hosts, Services, and Monitoring Features are displayed.



Tactical Monitoring Overview
Last Updated: Thu Jun 26 14:33:52 IST 2014
Updated every 90 seconds
Nagios® Core™ 3.5.1 - www.nagios.org
Logged in as *nagiosadmin*

Monitoring Performance	
Service Check Execution Time:	0.00 / 4.26 / 1.454 sec
Service Check Latency:	0.00 / 0.26 / 0.123 sec
Host Check Execution Time:	0.11 / 0.21 / 0.161 sec
Host Check Latency:	0.03 / 0.83 / 0.431 sec
# Active Host / Service Checks:	2 / 42
# Passive Host / Service Checks:	0 / 1

Network Outages
0 Outages

Network Health
Host Health:
Service Health:

Hosts			
0 Down	0 Unreachable	2 Up	0 Pending

Services				
2 Critical	1 Warning	1 Unknown	38 Ok	1 Pending
2 Unhandled Problems	1 Unhandled Problems	1 Unhandled Problems		1 Disabled

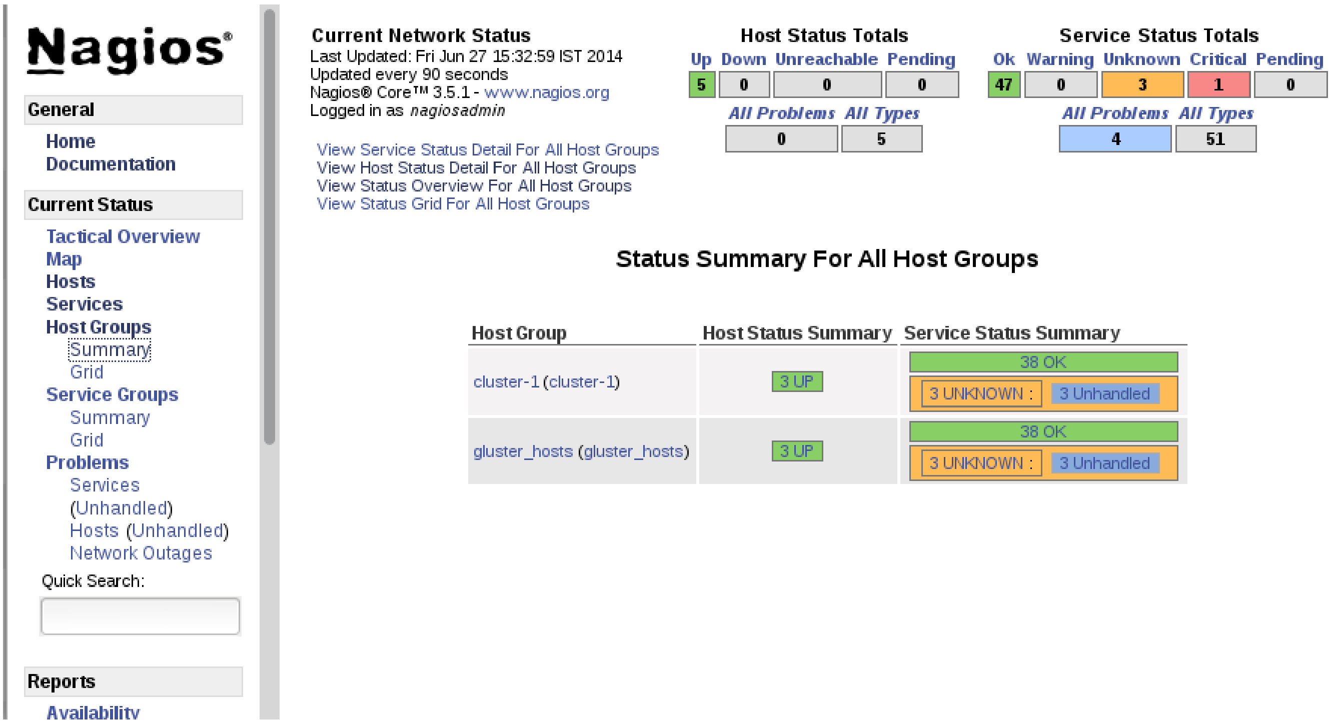
Monitoring Features

Flap Detection	Notifications	Event Handlers	Active Checks	Passive Checks
✓ All Services Enabled No Services Flapping 2 Hosts Disabled No Hosts Flapping	✓ All Services Enabled All Hosts Enabled	✓ All Services Enabled All Hosts Enabled	✓ 1 Service Disabled All Hosts Enabled	✓ All Services Enabled All Hosts Enabled

Figure 18.5. Tactical Overview

Host Status

To view the status summary of all the hosts, click **`Summary`** under **`Host Groups`** in the left pane.



Current Network Status
Last Updated: Fri Jun 27 15:32:59 IST 2014
Updated every 90 seconds
Nagios® Core™ 3.5.1 - www.nagios.org
Logged in as *nagiosadmin*

View Service Status Detail For All Host Groups
View Host Status Detail For All Host Groups
View Status Overview For All Host Groups
View Status Grid For All Host Groups

Host Status Totals

Up	Down	Unreachable	Pending
5	0	0	0

All Problems	All Types
0	5

Service Status Totals

Ok	Warning	Unknown	Critical	Pending
47	0	3	1	0

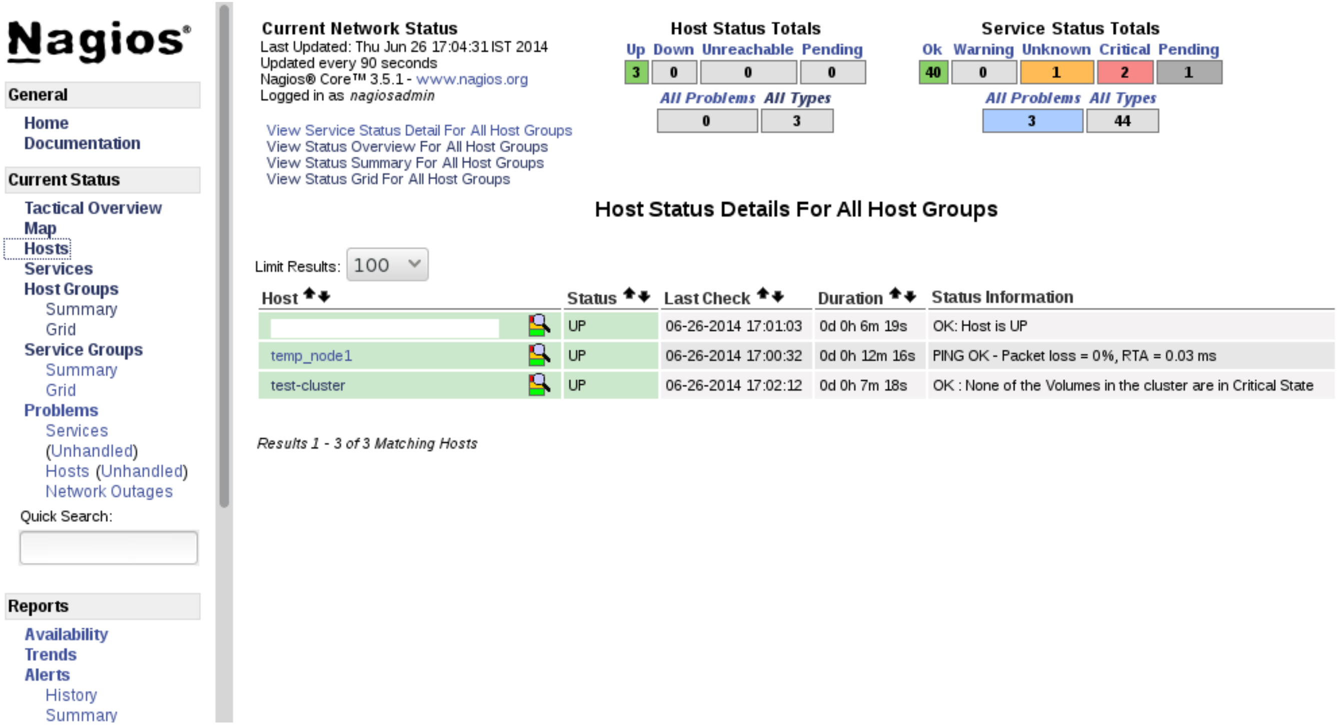
All Problems	All Types
4	51

Status Summary For All Host Groups

Host Group	Host Status Summary	Service Status Summary
cluster-1 (cluster-1)	3 UP	38 OK 3 UNKNOWN : 3 Unhandled
gluster_hosts (gluster_hosts)	3 UP	38 OK 3 UNKNOWN : 3 Unhandled

Figure 18.6. Host Groups Summary

To view the list of all hosts and their status, click **`Hosts`** in the left pane.



Current Network Status
Last Updated: Thu Jun 26 17:04:31 IST 2014
Updated every 90 seconds
Nagios® Core™ 3.5.1 - www.nagios.org
Logged in as *nagiosadmin*

Host Status Totals

Up	Down	Unreachable	Pending
3	0	0	0

All Problems	All Types
0	3

Service Status Totals

Ok	Warning	Unknown	Critical	Pending
40	0	1	2	1

All Problems	All Types
3	44

Host Status Details For All Host Groups

Host	Status	Last Check	Duration	Status Information
	UP	06-26-2014 17:01:03	0d 0h 6m 19s	OK: Host is UP
temp_node1	UP	06-26-2014 17:00:32	0d 0h 12m 16s	PING OK - Packet loss = 0%, RTA = 0.03 ms
test-cluster	UP	06-26-2014 17:02:12	0d 0h 7m 18s	OK : None of the Volumes in the cluster are in Critical State

Results 1 - 3 of 3 Matching Hosts

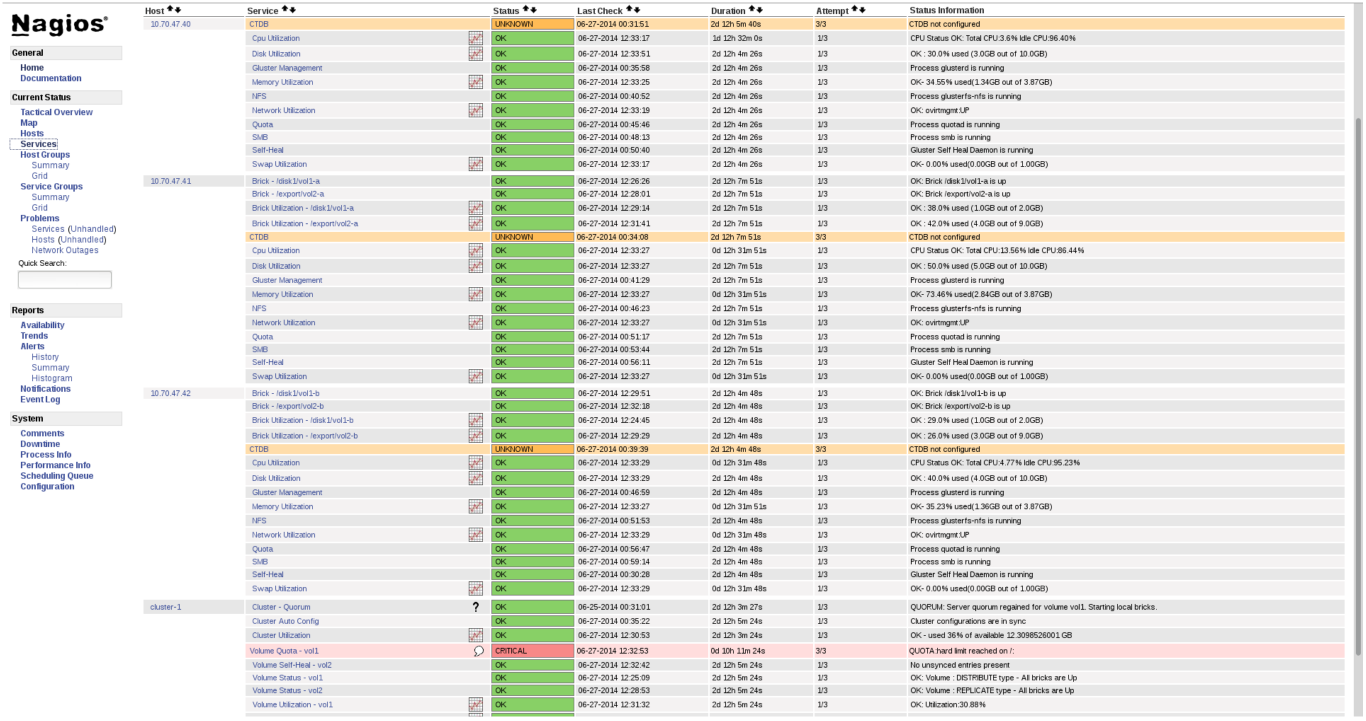
Figure 18.7. Host Status

Note

Cluster also will be shown as Host in Nagios and it will have all the volume services.

Service Status

To view the list of all hosts and their service status click **`Services`** in the left pane.



Host	Service	Status	Last Check	Duration	Attempt	Status Information
10.70.47.40	CTDB	UNKNOWN	06-27-2014 00:31:51	2d 12h 5m 40s	3/3	CTDB not configured
	Cpu Utilization	OK	06-27-2014 12:33:17	1d 12h 32m 0s	1/3	CPU Status OK: Total CPU:3.6% Idle CPU:96.40%
	Disk Utilization	OK	06-27-2014 12:33:51	2d 12h 4m 26s	1/3	OK : 30.0% used (3.0GB out of 10.0GB)
	Gluster Management	OK	06-27-2014 00:35:58	2d 12h 4m 26s	1/3	Process glusterd is running
	Memory Utilization	OK	06-27-2014 12:33:25	2d 12h 4m 26s	1/3	OK- 34.55% used(1.34GB out of 3.87GB)
	NFS	OK	06-27-2014 00:40:52	2d 12h 4m 26s	1/3	Process glusterfs-nfs is running
	Network Utilization	OK	06-27-2014 12:33:19	2d 12h 4m 26s	1/3	OK: ovirtmgmt:UP
	Quota	OK	06-27-2014 00:45:46	2d 12h 4m 26s	1/3	Process quotad is running
	SMB	OK	06-27-2014 00:48:13	2d 12h 4m 26s	1/3	Process smb is running
	Self-Heal	OK	06-27-2014 00:50:40	2d 12h 4m 26s	1/3	Gluster Self Heal Daemon is running
	Swap Utilization	OK	06-27-2014 12:33:17	2d 12h 4m 26s	1/3	OK- 0.00% used(0.00GB out of 1.00GB)
10.70.47.41	Brick - /disk1/vol1-a	OK	06-27-2014 12:26:26	2d 12h 7m 51s	1/3	OK: Brick /disk1/vol1-a is up
	Brick - /export/vol2-a	OK	06-27-2014 12:28:01	2d 12h 7m 51s	1/3	OK: Brick /export/vol2-a is up
	Brick Utilization - /disk1/vol1-a	OK	06-27-2014 12:29:14	2d 12h 7m 51s	1/3	OK : 38.0% used (1.0GB out of 2.0GB)
	Brick Utilization - /export/vol2-a	OK	06-27-2014 12:31:41	2d 12h 7m 51s	1/3	OK : 42.0% used (4.0GB out of 9.0GB)
	CTDB	UNKNOWN	06-27-2014 00:34:08	2d 12h 7m 51s	3/3	CTDB not configured
	Cpu Utilization	OK	06-27-2014 12:33:27	0d 12h 31m 51s	1/3	CPU Status OK: Total CPU:13.56% Idle CPU:86.44%
	Disk Utilization	OK	06-27-2014 12:33:27	2d 12h 7m 51s	1/3	OK : 50.0% used (5.0GB out of 10.0GB)
	Gluster Management	OK	06-27-2014 00:41:29	2d 12h 7m 51s	1/3	Process glusterd is running
	Memory Utilization	OK	06-27-2014 12:33:27	0d 12h 31m 51s	1/3	OK- 73.46% used(2.84GB out of 3.87GB)
	NFS	OK	06-27-2014 00:46:23	2d 12h 7m 51s	1/3	Process glusterfs-nfs is running
	Network Utilization	OK	06-27-2014 12:33:27	0d 12h 31m 51s	1/3	OK: ovirtmgmt:UP
	Quota	OK	06-27-2014 00:51:17	2d 12h 7m 51s	1/3	Process quotad is running
	SMB	OK	06-27-2014 00:53:44	2d 12h 7m 51s	1/3	Process smb is running
	Self-Heal	OK	06-27-2014 00:56:11	2d 12h 7m 51s	1/3	Gluster Self Heal Daemon is running
	Swap Utilization	OK	06-27-2014 12:33:27	0d 12h 31m 51s	1/3	OK- 0.00% used(0.00GB out of 1.00GB)
10.70.47.42	Brick - /disk1/vol1-b	OK	06-27-2014 12:29:51	2d 12h 4m 48s	1/3	OK: Brick /disk1/vol1-b is up
	Brick - /export/vol2-b	OK	06-27-2014 12:32:18	2d 12h 4m 48s	1/3	OK: Brick /export/vol2-b is up
	Brick Utilization - /disk1/vol1-b	OK	06-27-2014 12:24:45	2d 12h 4m 48s	1/3	OK : 29.0% used (1.0GB out of 2.0GB)
	Brick Utilization - /export/vol2-b	OK	06-27-2014 12:29:29	2d 12h 4m 48s	1/3	OK : 26.0% used (3.0GB out of 9.0GB)
	CTDB	UNKNOWN	06-27-2014 00:39:39	2d 12h 4m 48s	3/3	CTDB not configured
	Cpu Utilization	OK	06-27-2014 12:33:29	0d 12h 31m 48s	1/3	CPU Status OK: Total CPU:4.77% Idle CPU:95.23%
	Disk Utilization	OK	06-27-2014 12:33:29	2d 12h 4m 48s	1/3	OK : 40.0% used (4.0GB out of 10.0GB)
	Gluster Management	OK	06-27-2014 00:46:59	2d 12h 4m 48s	1/3	Process glusterd is running
	Memory Utilization	OK	06-27-2014 12:33:27	0d 12h 31m 51s	1/3	OK- 35.23% used(1.36GB out of 3.87GB)
	NFS	OK	06-27-2014 00:51:53	2d 12h 4m 48s	1/3	Process glusterfs-nfs is running
	Network Utilization	OK	06-27-2014 12:33:29	0d 12h 31m 48s	1/3	OK: ovirtmgmt:UP
	Quota	OK	06-27-2014 00:56:47	2d 12h 4m 48s	1/3	Process quotad is running
	SMB	OK	06-27-2014 00:59:14	2d 12h 4m 48s	1/3	Process smb is running
	Self-Heal	OK	06-27-2014 00:30:28	2d 12h 4m 48s	1/3	Gluster Self Heal Daemon is running
	Swap Utilization	OK	06-27-2014 12:33:29	0d 12h 31m 48s	1/3	OK- 0.00% used(0.00GB out of 1.00GB)
cluster-1	Cluster - Quorum	OK	06-25-2014 00:31:01	2d 12h 3m 27s	1/3	QUORUM: Server quorum regained for volume vol1. Starting local bricks.
	Cluster Auto Config	OK	06-27-2014 00:35:22	2d 12h 5m 24s	1/3	Cluster configurations are in sync
	Cluster Utilization	OK	06-27-2014 12:30:53	2d 12h 3m 24s	1/3	OK - used 36% of available 12.3098526001 GB
	Volume Quota - vol1	CRITICAL	06-27-2014 12:32:53	0d 10h 11m 24s	3/3	QUOTA:hard limit reached on /:
	Volume Self-Heal - vol2	OK	06-27-2014 12:32:42	2d 12h 5m 24s	1/3	No unsynced entries present
	Volume Status - vol1	OK	06-27-2014 12:25:09	2d 12h 5m 24s	1/3	OK: Volume : DISTRIBUTE type - All bricks are Up
	Volume Status - vol2	OK	06-27-2014 12:28:53	2d 12h 5m 24s	1/3	OK: Volume : REPLICATE type - All bricks are Up
	Volume Utilization - vol1	OK	06-27-2014 12:31:32	2d 12h 5m 24s	1/3	OK: Utilization:30.88%

Figure 18.8. Service Status

> **Note**
>
> In the left pane of Nagios Server GUI, click **Availability** and **Trends** under the **Reports** field to view the Host and Services Availability and Trends.

Host Services

1. Click **Hosts** in the left pane. The list of hosts are displayed.
2. Click corresponding to the host name to view the host details.
3. Select the service name to view the Service State Information. You can view the utilization of the following services:
   - Memory
   - Swap
   - CPU
   - Network
   - Brick
   - Disk

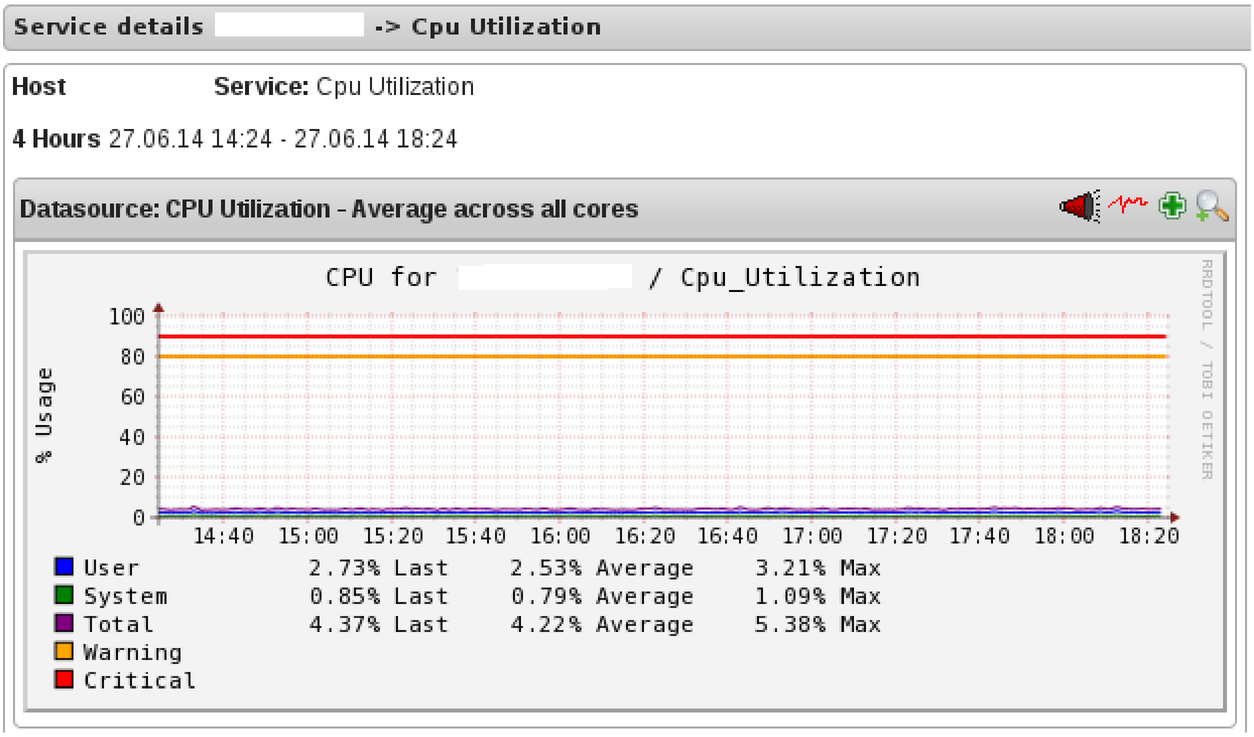
     The Brick/Disk Utilization Performance data has four sets of information for every mount point which are brick/disk space detail, inode detail of a brick/disk, thin pool utilization and thin pool metadata utilization if brick/disk is made up of thin LV.

     The Performance data for services is displayed in the following format: *value[UnitOfMeasurement]*;*warningthreshold*;*criticalthreshold*;*min*;*max*.

     For Example,

     Performance Data: /bricks/brick2=31.596%;80;90;0;0.990 /bricks/brick2.inode=0.003%;80;90;0;1048064 /bricks/brick2.thinpool=19.500%;80;90;0;1.500 /bricks/brick2.thinpool-metadata=4.100%;80;90;0;0.004

     As part of disk utilization service, the following mount points will be monitored: **/ , /boot, /home, /var and /usr** if available.
4. To view the utilization graph, click corresponding to the service name. The utilization graph is displayed.



Service details [redacted] **-> Cpu Utilization**

Host **Service:** Cpu Utilization

4 Hours 27.06.14 14:24 - 27.06.14 18:24

Datasource: CPU Utilization - Average across all cores

```
CPU for [redacted] / Cpu_Utilization
```

```
🟦 User       2.73% Last    2.53% Average    3.21% Max
🟩 System     0.85% Last    0.79% Average    1.09% Max
🟪 Total      4.37% Last    4.22% Average    5.38% Max
🟧 Warning
🟥 Critical
```

Figure 18.9. CPU Utilization

5. To monitor status, click on the service name. You can monitor the status for the following resources:
   - Disk
   - Network
6. To monitor process, click on the process name. You can monitor the following processes:
   - Gluster NFS (Network File System)
   - Self-Heal (Self-Heal)
   - Gluster Management (glusterd)
   - Quota (Quota daemon)
   - CTDB
   - SMB

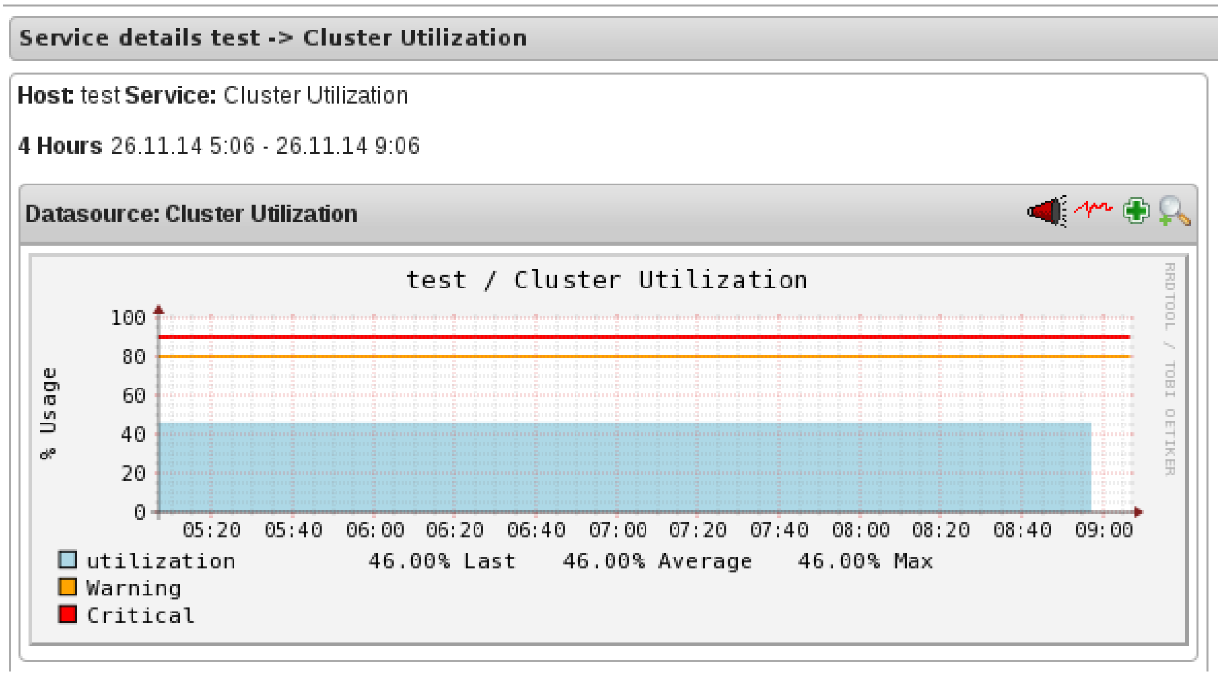
> **Note**
>
> Monitoring Openstack Swift operations is not supported.

Cluster Services

1. Click **`Hosts`** in the left pane. The list of hosts and clusters are displayed.
2. Click 🔍 corresponding to the cluster name to view the cluster details.

3. To view utilization graph, click corresponding to the service name. You can monitor the following utilizations:

   - Cluster
   - Volume



Service details test -> Cluster Utilization

Host: test **Service:** Cluster Utilization

4 Hours 26.11.14 5:06 - 26.11.14 9:06

Datasource: Cluster Utilization

Figure 18.10. Cluster Utilization

4. To monitor status, click on the service name. You can monitor the status for the following resources:

   - Host
   - Volume
   - Brick

5. To monitor cluster services, click on the service name. You can monitor the following:

   - Volume Quota
   - Volume Geo-replication
   - Volume Split-Brain
   - Cluster Quorum (A cluster quorum service would be present only when there are volumes in the cluster.)

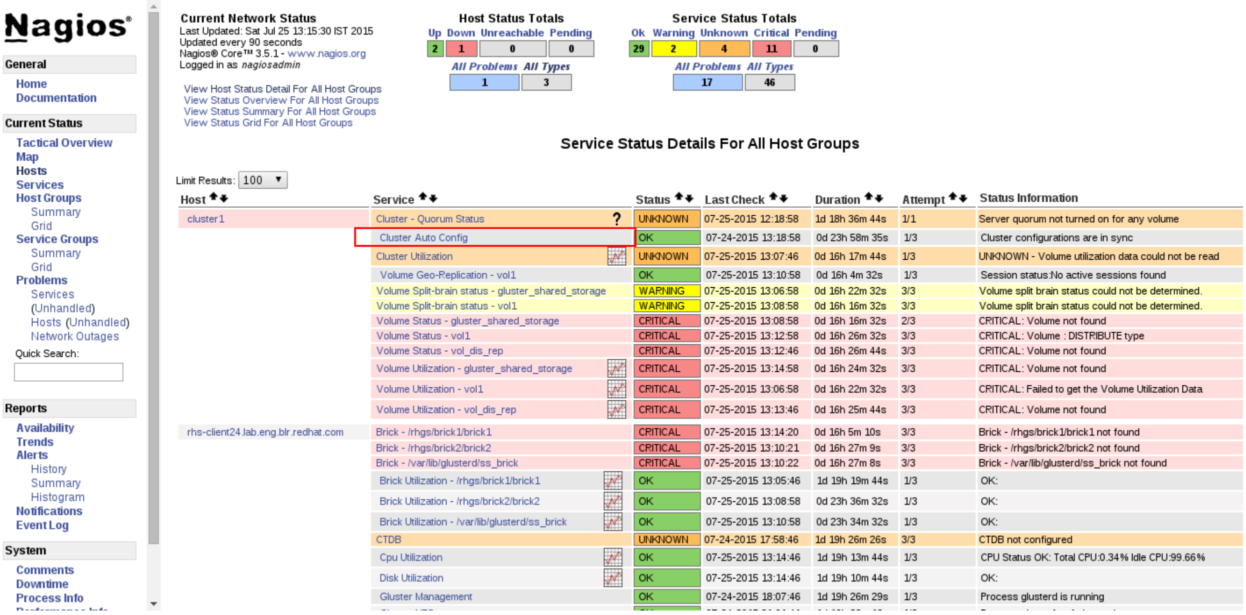
Rescheduling Cluster Auto config using Nagios Server GUI

If new nodes or services are added or removed, or if snapshot restore is performed on Red Hat Gluster Storage node, reschedule the **`Cluster Auto config`** service using Nagios Server GUI or execute the **`configure-gluster-nagios`** command. To synchronize the configurations using Nagios Server GUI, perform the steps given below:

1. Login to the Nagios Server GUI using the following URL in your browser with nagiosadmin user name and password.

```
https://NagiosServer-HostName-or-IPaddress/nagios
```

2. Click **Services** in left pane of Nagios server GUI and click **Cluster Auto Config**.



Current Network Status
Last Updated: Sat Jul 25 13:15:30 IST 2015
Updated every 90 seconds
Nagios® Core™ 3.5.1 - www.nagios.org
Logged in as *nagiosadmin*

View Host Status Detail For All Host Groups
View Status Overview For All Host Groups
View Status Summary For All Host Groups
View Status Grid For All Host Groups

Host Status Totals

Up	Down	Unreachable	Pending
2	1	0	0

All Problems	All Types
1	3

Service Status Totals

Ok	Warning	Unknown	Critical	Pending
29	2	4	11	0

All Problems	All Types
17	46

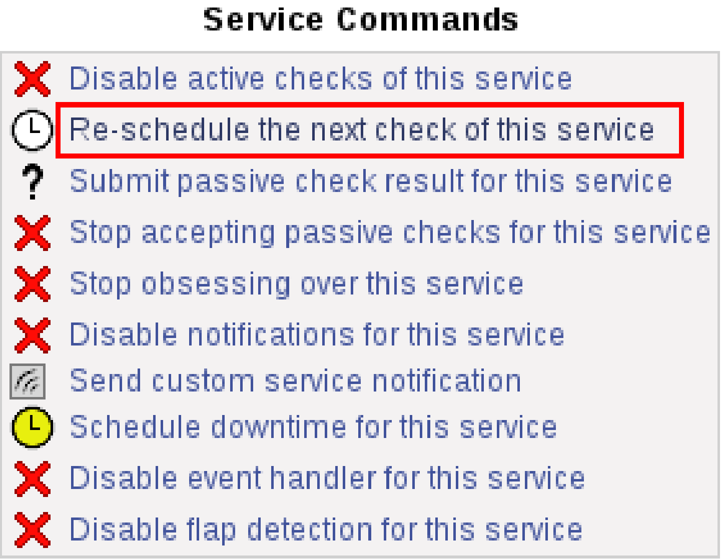
Service Status Details For All Host Groups

Limit Results: 100

Host	Service	Status	Last Check	Duration	Attempt	Status Information
cluster1	Cluster - Quorum Status	UNKNOWN	07-25-2015 12:18:58	1d 18h 36m 44s	1/1	Server quorum not turned on for any volume
	Cluster Auto Config	OK	07-24-2015 13:18:58	0d 23h 58m 35s	1/3	Cluster configurations are in sync
	Cluster Utilization	UNKNOWN	07-25-2015 13:07:46	0d 16h 17m 44s	1/3	UNKNOWN - Volume utilization data could not be read
	Volume Geo-Replication - vol1	OK	07-25-2015 13:10:58	0d 16h 4m 32s	1/3	Session status:No active sessions found
	Volume Split-brain status - gluster_shared_storage	WARNING	07-25-2015 13:06:58	0d 16h 22m 32s	3/3	Volume split brain status could not be determined.
	Volume Split-brain status - vol1	WARNING	07-25-2015 13:08:58	0d 16h 16m 32s	3/3	Volume split brain status could not be determined.
	Volume Status - gluster_shared_storage	CRITICAL	07-25-2015 13:08:58	0d 16h 16m 32s	2/3	CRITICAL: Volume not found
	Volume Status - vol1	CRITICAL	07-25-2015 13:12:58	0d 16h 26m 32s	3/3	CRITICAL: Volume : DISTRIBUTE type
	Volume Status - vol_dis_rep	CRITICAL	07-25-2015 13:12:46	0d 16h 26m 44s	3/3	CRITICAL: Volume not found
	Volume Utilization - gluster_shared_storage	CRITICAL	07-25-2015 13:14:58	0d 16h 24m 32s	3/3	CRITICAL: Volume not found
	Volume Utilization - vol1	CRITICAL	07-25-2015 13:06:58	0d 16h 22m 32s	3/3	CRITICAL: Failed to get the Volume Utilization Data
	Volume Utilization - vol_dis_rep	CRITICAL	07-25-2015 13:13:46	0d 16h 25m 44s	3/3	CRITICAL: Volume not found
rhs-client24.lab.eng.blr.redhat.com	Brick - /rhgs/brick1/brick1	CRITICAL	07-25-2015 13:14:20	0d 16h 5m 10s	3/3	Brick - /rhgs/brick1/brick1 not found
	Brick - /rhgs/brick2/brick2	CRITICAL	07-25-2015 13:10:21	0d 16h 27m 9s	3/3	Brick - /rhgs/brick2/brick2 not found
	Brick - /var/lib/glusterd/ss_brick	CRITICAL	07-25-2015 13:10:22	0d 16h 27m 8s	3/3	Brick - /var/lib/glusterd/ss_brick not found
	Brick Utilization - /rhgs/brick1/brick1	OK	07-25-2015 13:05:46	1d 19h 19m 44s	1/3	OK:
	Brick Utilization - /rhgs/brick2/brick2	OK	07-25-2015 13:08:58	0d 23h 36m 32s	1/3	OK:
	Brick Utilization - /var/lib/glusterd/ss_brick	OK	07-25-2015 13:10:58	0d 23h 34m 32s	1/3	OK:
	CTDB	UNKNOWN	07-24-2015 17:58:46	1d 19h 26m 26s	3/3	CTDB not configured
	Cpu Utilization	OK	07-25-2015 13:14:46	1d 19h 13m 44s	1/3	CPU Status OK: Total CPU:0.34% Idle CPU:99.66%
	Disk Utilization	OK	07-25-2015 13:14:46	1d 19h 10m 44s	1/3	OK:
	Gluster Management	OK	07-24-2015 18:07:46	1d 19h 26m 29s	1/3	Process glusterd is running

Figure 18.11. Nagios Services

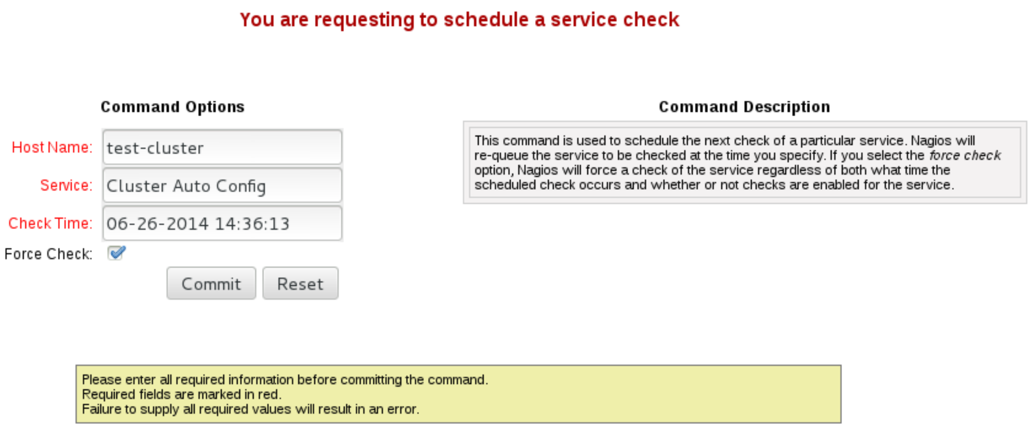
3. In **Service Commands**, click **Re-schedule the next check of this service**. The **Command Options** window is displayed.


Service Commands

Disable active checks of this service
Re-schedule the next check of this service
Submit passive check result for this service
Stop accepting passive checks for this service
Stop obsessing over this service
Disable notifications for this service
Send custom service notification
Schedule downtime for this service
Disable event handler for this service
Disable flap detection for this service

Figure 18.12. Service Commands

4. In **Command Options** window, click **Commit**.



You are requesting to schedule a service check

Command Options

Host Name: test-cluster

Service: Cluster Auto Config

Check Time: 06-26-2014 14:36:13

Force Check: ☑

Command Description

This command is used to schedule the next check of a particular service. Nagios will re-queue the service to be checked at the time you specify. If you select the *force check* option, Nagios will force a check of the service regardless of both what time the scheduled check occurs and whether or not checks are enabled for the service.

Please enter all required information before committing the command.
Required fields are marked in red.
Failure to supply all required values will result in an error.

Figure 18.13. Command Options

Enabling and Disabling Notifications using Nagios GUI

You can enable or disable Host and Service notifications through Nagios GUI.

- To enable and disable Host Notifcations:
  - Login to the Nagios Server GUI using the following URL in your browser with **nagiosadmin** user name and password.

    ```
    https://NagiosServer-HostName-or-IPaddress/nagios
    ```

  - Click **Hosts** in left pane of Nagios server GUI and select the host.
  - Click **Enable notifications for this host** or **Disable notifications for this host** in Host Commands section.
  - Click **Commit** to enable or disable notification for the selected host.
- To enable and disable Service Notification:
  - Login to the Nagios Server GUI.
  - Click **Services** in left pane of Nagios server GUI and select the service to enable or disable.
  - Click **Enable notifications for this service** or **Disable notifications for this service** from the Service Commands section.
  - Click **Commit** to enable or disable the selected service notification.
- To enable and disable all Service Notifications for a host:
  - Login to the Nagios Server GUI.
  - Click **Hosts** in left pane of Nagios server GUI and select the host to enable or disable all services notifications.
  - Click **Enable notifications for all services on this host** or **Disable notifications for all services on this host** from the Service Commands section.
  - Click **Commit** to enable or disable all service notifications for the selected host.
- To enable or disable all Notifications:

- Login to the Nagios Server GUI.
- Click **Process Info** under **Systems** section from left pane of Nagios server GUI.
- Click **Enable notifications** or **Disable notifications** in Process Commands section.
- Click **Commit**.

Enabling and Disabling Service Monitoring using Nagios GUI

You can enable a service to monitor or disable a service you have been monitoring using the Nagios GUI.

- To enable Service Monitoring:
  - Login to the Nagios Server GUI using the following URL in your browser with **nagiosadmin** user name and password.

    ```
    https://NagiosServer-HostName-or-IPaddress/nagios
    ```

  - Click **Services** in left pane of Nagios server GUI and select the service to enable monitoring.
  - Click **Enable active checks of this service** from the Service Commands and click **Commit**.
  - Click **Start accepting passive checks for this service** from the Service Commands and click **Commit**.

    Monitoring is enabled for the selected service.

- To disable Service Monitoring:
  - Login to the Nagios Server GUI using the following URL in your browser with **nagiosadmin** user name and password.

    ```
    https://NagiosServer-HostName-or-IPaddress/nagios
    ```

  - Click **Services** in left pane of Nagios server GUI and select the service to disable monitoring.
  - Click **Disable active checks of this service** from the Service Commands and click **Commit**.
  - Click **Stop accepting passive checks for this service** from the Service Commands and click **Commit**.

    Monitoring is disabled for the selected service.

Monitoring Services Status and Messages

Note

Nagios sends email and SNMP notifications, once a service status changes. Refer *Configuring Nagios Server to Send Mail Notifications* section of *Red Hat Gluster Storage 3 Console Administation Guide* to configure email notification and *Configuring Simple Network Management Protocol (SNMP) Notification* section of *Red Hat Gluster Storage 3 Administation Guide* to configure SNMP notification.

Table 18.1.

Service Name	Status	Messsage	Description
SMB	OK	OK: No gluster volume uses smb	When no volumes are exported through smb.
	OK	Process smb is running	When SMB service is running and when volumes are exported using SMB.
	CRITICAL	CRITICAL: Process smb is not running	When SMB service is down and one or more volumes are exported through SMB.
CTDB	UNKNOWN	CTDB not configured	When CTDB service is not running, and smb or nfs service is running.
	CRITICAL	Node status: BANNED/STOPPED	When CTDB service is running but Node status is *BANNED/STOPPED*.
	WARNING	Node status: UNHEALTHY/DISABLED /PARTIALLY_ONLINE	When CTDB service is running but Node status is *UNHEALTHY/DISABLED /PARTIALLY_ONLINE*.
	OK	Node status: OK	When CTDB service is running and healthy.
Gluster Management	OK	Process glusterd is running	When glusterd is running as unique.
	WARNING	PROCS WARNING: 3 processes	When there are more then one glusterd is running.
	CRITICAL	CRITICAL: Process glusterd is not running	When there is no glusterd process running.
	UNKNOWN	NRPE: Unable to read output	When unable to communicate or read output
Gluster NFS	OK	OK: No gluster volume uses nfs	When no volumes are configured to be exported through NFS.
	OK	Process glusterfs-nfs is running	When glusterfs-nfs process is running.
	CRITICAL	CRITICAL: Process glusterfs-nfs is not running	When glusterfs-nfs process is down and there are volumes which requires NFS export.

Service Name	Status	Messsage	Description
Auto-Config	OK	Cluster configurations are in sync	When auto-config has not detected any change in Gluster configuration. This shows that Nagios configuration is already in synchronization with the Gluster configuration and auto-config service has not made any change in Nagios configuration.
	OK	Cluster configurations synchronized successfully from host *host-address*	When auto-config has detected change in the Gluster configuration and has successfully updated the Nagios configuration to reflect the change Gluster configuration.
	CRITICAL	Can't remove all hosts except sync host in 'auto' mode. Run auto discovery manually.	When the host used for auto-config itself is removed from the Gluster peer list. Auto-config will detect this as all host except the synchronized host is removed from the cluster. This will not change the Nagios configuration and the user need to manually run the auto-config.
QUOTA	OK	OK: Quota not enabled	When quota is not enabled in any volumes.
	OK	Process quotad is running	When glusterfs-quota service is running.
	CRITICAL	CRITICAL: Process quotad is not running	When glusterfs-quota service is down and quota is enabled for one or more volumes.
CPU Utilization	OK	CPU Status OK: Total CPU:4.6% Idle CPU:95.40%	When CPU usage is less than 80%.
	WARNING	CPU Status WARNING: Total CPU:82.40% Idle CPU:17.60%	When CPU usage is more than 80%.
	CRITICAL	CPU Status CRITICAL: Total CPU:97.40% Idle CPU:2.6%	When CPU usage is more than 90%.
Memory Utilization	OK	OK- 65.49% used(1.28GB out of 1.96GB)	When used memory is below warning threshold. (Default warning threshold is 80%)

Service Name	Status	Messsage	Description
	WARNING	WARNING- 85% used(1.78GB out of 2.10GB)	When used memory is below critical threshold (Default critical threshold is 90%) and greater than or equal to warning threshold (Default warning threshold is 80%).
	CRITICAL	CRITICAL- 92% used(1.93GB out of 2.10GB)	When used memory is greater than or equal to critical threshold (Default critical threshold is 90%)
Brick Utilization	OK	OK	When used space of any of the four parameters, space detail, inode detail, thin pool, and thin pool-metadata utilizations, are below threshold of 80%.
	WARNING	WARNING:mount point /brick/brk1 Space used (0.857 / 1.000) GB	If any of the four parameters, space detail, inode detail, thin pool utilization, and thinpool-metadata utilization, crosses warning threshold of 80% (Default is 80%).
	CRITICAL	CRITICAL : mount point /brick/brk1 (inode used 9980/1000)	If any of the four parameters, space detail, inode detail, thin pool utilization, and thinpool-metadata utilizations, crosses critical threshold 90% (Default is 90%).
Disk Utilization	OK	OK	When used space of any of the four parameters, space detail, inode detail, thin pool utilization, and thinpool-metadata utilizations, are below threshold of 80%.
	WARNING	WARNING:mount point /boot Space used (0.857 / 1.000) GB	When used space of any of the four parameters, space detail, inode detail, thin pool utilization, and thinpool-metadata utilizations, are above warning threshold of 80%.

Service Name	Status	Messsage	Description
	CRITICAL	CRITICAL : mount point /home (inode used 9980/1000)	If any of the four parameters, space detail, inode detail, thin pool utilization, and thinpool-metadata utilizations, crosses critical threshold 90% (Default is 90%).
Network Utilization	OK	OK: tun0:UP,wlp3s0:UP,virbr0:UP	When all the interfaces are UP.
	WARNING	WARNING: tun0:UP,wlp3s0:UP,virbr0:DOWN	When any of the interfaces is down.
	UNKNOWN	UNKNOWN	When network utilization/status is unknown.
Swap Utilization	OK	OK- 0.00% used(0.00GB out of 1.00GB)	When used memory is below warning threshold (Default warning threshold is 80%).
	WARNING	WARNING- 83% used(1.24GB out of 1.50GB)	When used memory is below critical threshold (Default critical threshold is 90%) and greater than or equal to warning threshold (Default warning threshold is 80%).
	CRITICAL	CRITICAL- 83% used(1.42GB out of 1.50GB)	When used memory is greater than or equal to critical threshold (Default critical threshold is 90%).
Cluster Quorum	PENDING		When cluster.server-quorum-type is not set to *server*; or when there are no problems in the cluster identified.
	OK	Quorum regained for volume	When quorum is regained for volume.
	CRITICAL	Quorum lost for volume	When quorum is lost for volume.
Volume Geo-replication	OK	"Session Status: *slave_vol1*-OK .....*slave_voln*-OK.	When all sessions are active.
	OK	Session status :No active sessions found	When Geo-replication sessions are deleted.
	CRITICAL	Session Status: *slave_vol1*-FAULTY *slave_vol2*-OK	If one or more nodes are Faulty and there's no replica pair that's active.

Service Name	Status	Messsage	Description
	WARNING	Session Status: *slave_vol1*-NOT_STARTED *slave_vol2*-STOPPED *slave_vol3*-PARTIAL_FAULTY	- Partial faulty state occurs with replicated and distributed replicate volume when one node is faulty, but the replica pair is active. - *STOPPED* state occurs when Geo-replication sessions are stopped. - *NOT_STARTED* state occurs when there are multiple Geo-replication sessions and one of them is stopped.
	WARNING	Geo replication status could not be determined.	When there's an error in getting Geo replication status. This error occurs when **`volfile`** is locked as another transaction is in progress.
	UNKNOWN	Geo replication status could not be determined.	When glusterd is down.
Volume Quota	OK	QUOTA: not enabled or configured	When quota is not set
	OK	QUOTA:OK	When quota is set and usage is below quota limits.
	WARNING	QUOTA:Soft limit exceeded on *path of directory*	When quota exceeds soft limit.
	CRITICAL	QUOTA:hard limit reached on *path of directory*	When quota reaches hard limit.
	UNKNOWN	QUOTA: Quota status could not be determined as command execution failed	When there's an error in getting Quota status. This occurs when - Volume is stopped or glusterd service is down. - volfile is locked as another transaction in progress.
Volume Status	OK	Volume : *volume type* - All bricks are Up	When all volumes are up.

Service Name	Status	Messsage	Description
	WARNING	Volume :*volume type* Brick(s) - *list of bricks* is|are down, but replica pair(s) are up	When bricks in the volume are down but replica pairs are up.
	UNKNOWN	Command execution failed *Failure message*	When command execution fails.
	CRITICAL	Volume not found.	When volumes are not found.
	CRITICAL	Volume: *volume-type* is stopped.	When volumes are stopped.
	CRITICAL	Volume : *volume type* - All bricks are down.	When all bricks are down.
	CRITICAL	Volume : *volume type* Bricks - *brick list* are down, along with one or more replica pairs	When bricks are down along with one or more replica pairs.
Volume Self-Heal (available in Red Hat Gluster Storage version 3.1.0 and earlier)	OK		When volume is not a replicated volume, there is no self-heal to be done.
	OK	No unsynced entries present	When there are no unsynched entries in a replicated volume.
	WARNING	Unsynched entries present : There are unsynched entries present.	If self-heal process is turned on, these entries may be auto healed. If not, self-heal will need to be run manually. If unsynchronized entries persist over time, this could indicate a split brain scenario.
	WARNING	Self heal status could not be determined as the volume was deleted	When self-heal status can not be determined as the volume is deleted.
	UNKNOWN		When there's an error in getting self heal status. This error occurs when: - Volume is stopped or glusterd service is down. - volfile is locked as another transaction in progress.
Volume Self-Heal Info (available in Red Hat Gluster Storage version 3.1.3 and later)	OK	No unsynced entries found.	Displayed when there are no entries in a replicated volume that haven't been synced.

Service Name	Status	Messsage	Description
	WARNING	Unsynced entries found.	Displayed when there are entries in a replicated volume that still need to be synced. If self-heal is enabled, these may heal automatically. If self-heal is not enabled, healing must be run manually.
	WARNING	Volume heal information could not be determined.	Displayed when self-heal status cannot be determined, usually because the volume has been deleted.
	UNKNOWN	Glusterd cannot be queried.	Displayed when self-heal status cannot be retrieved. usually because the volume has been stopped, the glusterd service is down, or the volfile is locked because another transaction is in progress.
Volume Split-Brain Status (available in Red Hat Gluster Storage version 3.1.1 and later)	OK	No split-brain entries found.	Displayed when files are present and do not have split-brain issues.
	UNKNOWN	Glusterd cannot be queried.	Displayed when split-brain status cannot be retrieved, usually because the volume has been stopped, the glusterd service is down, or the volfile is locked because another transaction is in progress.
	WARNING	Volume split-brain status could not be determined.	Displayed when split-brain status cannot be determined, usually because the volume no longer exists.
	CRITICAL	*14* entries found in split-brain state.	Displays the number of files in a split-brain state when files in split-brain state are detected.
Cluster Utilization	OK	OK : 28.0% used (1.68GB out of 6.0GB)	When used % is below the warning threshold (Default warning threshold is 80%).

Service Name	Status	Messsage	Description
Volume Utilization	WARNING	WARNING: 82.0% used (4.92GB out of 6.0GB)	Used% is above the warning limit. (Default warning threshold is 80%)
	CRITICAL	CRITICAL : 92.0% used (5.52GB out of 6.0GB)	Used% is above the warning limit. (Default critical threshold is 90%)
	UNKNOWN	Volume utilization data could not be read	When volume services are present, but the volume utilization data is not available as it's either not populated yet or there is error in fetching volume utilization data.
	OK	OK: Utilization: 40 %	When used % is below the warning threshold (Default warning threshold is 80%).
	WARNING	WARNING - used *84%* of available *200* GB	When used % is above the warning threshold (Default warning threshold is 80%).
	CRITICAL	CRITICAL - used *96%* of available *200* GB	When used % is above the critical threshold (Default critical threshold is 90%).
	UNKNOWN	UNKNOWN - Volume utilization data could not be read	When all the bricks in the volume are killed or if glusterd is stopped in all the nodes in a cluster.

18.4. Monitoring Notifications

18.4.1. Configuring Nagios Server to Send Mail Notifications

1. In the **/etc/nagios/gluster/gluster-contacts.cfg** file, add contacts to send mail in the format shown below:

   Modify **contact_name**, **alias**, and **email**.

```
define contact {
        contact_name                                Contact1
        alias                                       ContactNameAlias
        email                                       email-address
        service_notification_period                 24x7
        service_notification_options                w,u,c,r,f,s
        service_notification_commands               notify-service-by-
email
        host_notification_period                    24x7
        host_notification_options                   d,u,r,f,s
        host_notification_commands                  notify-host-by-email
}
```

```
define contact {
        contact_name                            Contact2
        alias                                   ContactNameAlias2
        email                                   email-address
        service_notification_period             24x7
        service_notification_options            w,u,c,r,f,s
        service_notification_commands           notify-service-by-
email
        host_notification_period                24x7
        host_notification_options               d,u,r,f,s
        host_notification_commands              notify-host-by-email
}
```

The **service_notification_options** directive is used to define the service states for which notifications can be sent out to this contact. Valid options are a combination of one or more of the following:

- **w**: Notify on WARNING service states
- **u**: Notify on UNKNOWN service states
- **c**: Notify on CRITICAL service states
- **r**: Notify on service RECOVERY (OK states)
- **f**: Notify when the service starts and stops FLAPPING
- **n (none)**: Do not notify the contact on any type of service notifications

The **host_notification_options** directive is used to define the host states for which notifications can be sent out to this contact. Valid options are a combination of one or more of the following:

- **d**: Notify on DOWN host states
- **u**: Notify on UNREACHABLE host states
- **r**: Notify on host RECOVERY (UP states)
- **f**: Notify when the host starts and stops FLAPPING
- **s**: Send notifications when host or service scheduled downtime starts and ends
- **n (none)**: Do not notify the contact on any type of host notifications.

> **Note**
>
> By default, a contact and a contact group are defined for administrators in **contacts.cfg** and all the services and hosts will notify the administrators. Add suitable email id for administrator in **contacts.cfg** file.

2. To add a group to which the mail need to be sent, add the details as given below:

```
  define contactgroup{
        contactgroup_name                   Group1
        alias                               GroupAlias
        members                             Contact1,Contact2
}
```

3. In the **/etc/nagios/gluster/gluster-templates.cfg** file specify the contact name and contact group name for the services for which the notification need to be sent, as shown below:

   Add **contact_groups** name and **contacts** name.

```
define host{
   name                         gluster-generic-host
   use                          linux-server
   notifications_enabled        1
   notification_period          24x7
   notification_interval        120
   notification_options         d,u,r,f,s
   register                     0
   contact_groups         Group1
   contacts                     Contact1,Contact2
   }

 define service {
   name                         gluster-service
   use                          generic-service
   notifications_enabled       1
   notification_period          24x7
   notification_options         w,u,c,r,f,s
   notification_interval        120
   register                     0
   _gluster_entity              Service
   contact_groups        Group1
   contacts                 Contact1,Contact2

}
```

   You can configure notification for individual services by editing the corresponding node configuration file. For example, to configure notification for brick service, edit the corresponding node configuration file as shown below:

```
define service {
  use                             brick-service
  _VOL_NAME                       VolumeName
  __GENERATED_BY_AUTOCONFIG       1
  notes                           Volume : VolumeName
  host_name                       RedHatStorageNodeName
  _BRICK_DIR                      brickpath
  service_description             Brick Utilization - brickpath
  contact_groups          Group1
    contacts                Contact1,Contact2
}
```

4. To receive detailed information on every update when Cluster Auto-Config is run, edit **/etc/nagios/objects/commands.cfg** file add **$NOTIFICATIONCOMMENT$\n** after **$SERVICEOUTPUT$\n** option in **notify-service-by-email** and **notify-host-by-email**command definition as shown below:

```
# 'notify-service-by-email' command definition
define command{
        command_name    notify-service-by-email
        command_line    /usr/bin/printf "%b" "***** Nagios
*****\n\nNotification Type: $NOTIFICATIONTYPE$\n\nService:
$SERVICEDESC$\nHost: $HOSTALIAS$\nAddress: $HOSTADDRESS$\nState:
$SERVICESTATE$\n\nDate/Time: $LONGDATETIME$\n\nAdditional
Info:\n\n$SERVICEOUTPUT$\n $NOTIFICATIONCOMMENT$\n" | /bin/mail -s "**
$NOTIFICATIONTYPE$ Service Alert: $HOSTALIAS$/$SERVICEDESC$ is
$SERVICESTATE$ **" $CONTACTEMAIL$
        }
```

5. Restart the Nagios server using the following command:

```
# service nagios restart
```

The Nagios server sends notifications during status changes to the mail addresses specified in the file.

Note

- By default, the system ensures three occurences of the event before sending mail notifications.
- By default, Nagios Mail notification is sent using **/bin/mail** command. To change this, modify the definition for **notify-host-by-email** command and **notify-service-by-email** command in **/etc/nagios/objects/commands.cfg** file and configure the mail server accordingly.
- Ensure that the mail server is setup and configured.

18.4.2. Configuring Simple Network Management Protocol (SNMP) Notification

1. Log in as *root* user.

2. In the **/etc/nagios/gluster/snmpmanagers.conf** file, specify the Host Name or IP address and community name of the SNMP managers to whom the SNMP traps need to be sent as shown below:

```
HostName-or-IP-address public
```

In the **/etc/nagios/gluster/gluster-contacts.cfg** file specify the contacts name as *+snmp* as shown below:

```
define contact {
       contact_name                   snmp
       alias                          Snmp Traps
       email                          admin@ovirt.com
       service_notification_period    24x7
       service_notification_options   w,u,c,r,f,s
       service_notification_commands  gluster-notify-service-by-snmp
       host_notification_period       24x7
       host_notification_options      d,u,r,f,s
       host_notification_commands     gluster-notify-host-by-snmp
}
```

You can download the required Management Information Base (MIB) files from the URLs given below:

- NAGIOS-NOTIFY-MIB: https://github.com/nagios-plugins/nagios-mib/blob/master/MIB/NAGIOS-NOTIFY-MIB
- NAGIOS-ROOT-MIB: https://github.com/nagios-plugins/nagios-mib/blob/master/MIB/NAGIOS-ROOT-MIB

3. Restart Nagios using the following command:

```
# service nagios restart
```

18.5. Nagios Advanced Configuration

18.5.1. Creating Nagios User

To create a new Nagios user and set permissions for that user, follow the steps given below:

1. Login as **root** user.

2. Run the command given below with the new user name and type the password when prompted.

```
#  htpasswd /etc/nagios/passwd newUserName
```

3. Add permissions for the new user in **/etc/nagios/cgi.cfg** file as shown below:

```
authorized_for_system_information=nagiosadmin,newUserName
authorized_for_configuration_information=nagiosadmin,newUserName
authorized_for_system_commands=nagiosadmin,newUserName
authorized_for_all_services=nagiosadmin,newUserName
authorized_for_all_hosts=nagiosadmin,newUserName
authorized_for_all_service_commands=nagiosadmin,newUserName
authorized_for_all_host_commands=nagiosadmin,newUserName
```

Note

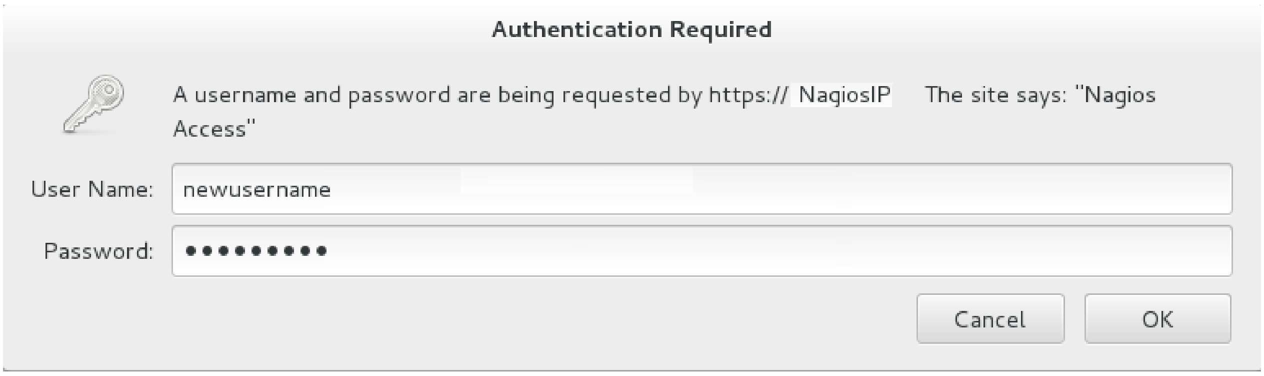
To set **read only** permission for users, add **authorized_for_read_only=*username*** in the **/etc/nagios/cgi.cfg** file.

4. Start **nagios** and **httpd** services using the following commands:

```
#  service httpd restart
#  service nagios restart
```

5. Verify Nagios access by using the following URL in your browser, and using the user name and password.

```
https://NagiosServer-HostName-or-IPaddress/nagios
```



Authentication Required

A username and password are being requested by https:// NagiosIP The site says: "Nagios Access"

User Name: newusername

Password: ••••••••••

Figure 18.14. Nagios Login

18.5.2. Changing Nagios Password

The default Nagios user name and password is **`nagiosadmin`**. This value is available in the **`/etc/nagios/cgi.cfg`** file.

1. Login as **`root`** user.

2. To change the default password for the Nagios Administrator user, run the following command with the new password:

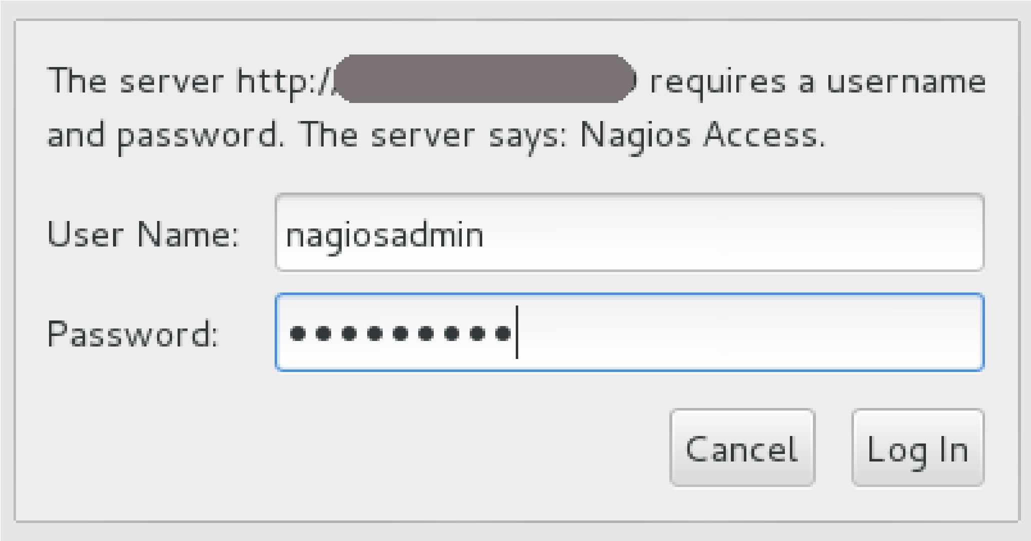
```
# htpasswd -c /etc/nagios/passwd nagiosadmin
```

3. Start **`nagios`** and **`httpd`** services using the following commands:

```
#  service httpd restart
#  service nagios restart
```

4. Verify Nagios access by using the following URL in your browser, and using the user name and password that was set in Step 2:

```
https://NagiosServer-HostName-or-IPaddress/nagios
```



The server http:/[redacted] requires a username and password. The server says: Nagios Access.

User Name: nagiosadmin

Password: ••••••••••

Figure 18.15. Nagios Login

18.5.3. Configuring SSL

For secure access of Nagios URL, configure SSL:

1. Create a 1024 bit RSA key using the following command:

   ```
   openssl genrsa -out /etc/ssl/private/{cert-file-name.key} 1024
   ```

2. Create an SSL certificate for the server using the following command:

   ```
   openssl req -key nagios-ssl.key -new | openssl x509 -out nagios-
   ssl.crt -days 365 -signkey  nagios-ssl.key -req
   ```

   Enter the server's host name which is used to access the Nagios Server GUI as *Common Name*.

3. Edit the **`/etc/httpd/conf.d/ssl.conf`** file and add path to SSL Certificate and key files correspondingly for **`SSLCertificateFile`** and **`SSLCertificateKeyFile`** fields as shown below:

   ```
    SSLCertificateFile     /etc/pki/tls/certs/nagios-ssl.crt
    SSLCertificateKeyFile  /etc/pki/tls/private/nagios-ssl.key
   ```

4. Edit the **`/etc/httpd/conf/httpd.conf`** file and comment the port 80 listener as shown below:

   ```
   # Listen 80
   ```

5. In **`/etc/httpd/conf/httpd.conf`** file, ensure that the following line is not commented:

   ```
   <Directory "/var/www/html">
   ```

6. Restart the **`httpd`** service on the **`nagios`** server using the following command:

   ```
   # service httpd restart
   ```

18.5.4. Integrating LDAP Authentication with Nagios

You can integrate LDAP authentication with Nagios plug-in. To integrate LDAP authentication, follow the steps given below:

1. In apache configuration file **`/etc/httpd/conf/httpd.conf`**, ensure that LDAP is installed and LDAP apache module is enabled.

   The configurations are displayed as given below if the LDAP apache module is enabled.You can enable the LDAP apache module by deleting the # symbol.

   ```
   LoadModule ldap_module modules/mod_ldap.so
   LoadModule authnz_ldap_module modules/mod_authnz_ldap.so
   ```

2. Edit the **`nagios.conf`** file in **`/etc/httpd/conf.d/nagios.conf`** with the corresponding values for the following:

   - AuthBasicProvider

   - AuthLDAPURL

   - AuthLDAPBindDN
   - AuthLDAPBindPassword

3. Edit the CGI authentication file **/etc/nagios/cgi.cfg** as given below with the path where Nagios is installed.

   ```
   nagiosinstallationdir = /usr/local/nagios/ or /etc/nagios/
   ```

4. Uncomment the lines shown below by deleting # and set permissions for specific users:

   > **Note**
   >
   > Replace **nagiosadmin** and *user names* with * to give any LDAP user full functionality of Nagios.

   ```
   authorized_for_system_information=user1,user2,user3

   authorized_for_configuration_information=nagiosadmin,user1,user2,user3

   authorized_for_system_commands=nagiosadmin,user1,user2,user3

   authorized_for_all_services=nagiosadmin,user1,user2,user3

   authorized_for_all_hosts=nagiosadmin,user1,user2,user3

   authorized_for_all_service_commands=nagiosadmin,user1,user2,user3

   authorized_for_all_host_commands=nagiosadmin,user1,user2,user3
   ```

5. Enable the **httpd_can_connect_ldap** boolean, if not enabled.

   ```
   # getsebool httpd_can_connect_ldap
   # setsebool httpd_can_connect_ldap on
   ```

6. Restart **httpd** service and **nagios** server using the following commands:

   ```
   # service httpd restart
   # service nagios restart
   ```

18.6. Configuring Nagios Manually

You can configure the Nagios server and node manually to monitor a Red Hat Gluster Storage trusted storage pool.

> **Note**
>
> It is recommended to configure Nagios using Auto-Discovery. For more information on configuring Nagios using Auto-Discovery, see Section 18.3.1, “Configuring Nagios”

For more information on Nagios Configuration files, see Chapter 22, *Nagios Configuration Files*

Configuring Nagios Server

1. In the **`/etc/nagios/gluster`** directory, create a directory with the cluster name. All configurations for the cluster are added in this directory.

2. In the **`/etc/nagios/gluster/`*`cluster-name`*** directory, create a file with name ***`clustername`*`.cfg`** to specify the **`host`** and **`hostgroup`** configurations. The service configurations for all the cluster and volume level services are added in this file.

   Note

   Cluster is configured as host and host group in Nagios.

   In the ***`clustername`*`.cfg`** file, add the following definitions:

   a. Define a host group with cluster name as shown below:

   ```
   define hostgroup{
                hostgroup_name                         cluster-name
                alias                                  cluster-name
       }
   ```

   b. Define a host with cluster name as shown below:

   ```
    define host{
               host_name                               cluster-name
               alias                                   cluster-name
               use                                     gluster-cluster
               address                                 cluster-name
       }
   ```

   c. Define *Cluster-Quorum* service to monitor cluster quorum status as shown below:

   ```
   define service {
                service_description                Cluster - Quorum
                use                                gluster-passive-
   service
           host_name                            cluster-name
       }
   ```

   d. Define the *Cluster Utilization* service to monitor cluster utilization as shown below:

   ```
   define service {
                service_description                  Cluster
   Utilization
                use gluster-service-with-graph
                check_command     check_cluster_vol_usage!warning-
   threshold!critcal-threshold;
                host_name                            cluster-name
       }
   ```

e. Add the following service definitions for each volume in the cluster:

- Volume Status service to monitor the status of the volume as shown below:

```
define service {
                 service_description             Volume
Status - volume-name
                 host_name                       cluster-
name
                 use gluster-service-without-graph
                 _VOL_NAME                       volume-
name
                 notes                           Volume
type : Volume-Type
                 check_command     check_vol_status!cluster-
name!volume-name
        }
```

- Volume Utilization service to monitor the volume utilization as shown below:

```
define service {
                 service_description             Volume
Utilization - volume-name
                 host_name                       cluster-
name
                 use gluster-service-with-graph
                 _VOL_NAME                       volume-
name
                 notes                           Volume
type : Volume-Type
                 check_command
check_vol_utilization!cluster-name!volume-name!warning-
threshold!critcal-threshold
        }
```

- Volume Split-brain service to monitor split brain status as shown below:

```
define service {
                         service_description    Volume
Split-brain status - volume-name
                         host_name                 cluster-
name
                         use gluster-service-without-graph
                         _VOL_NAME
volume-name
                        check_command
check_vol_heal_status!cluster1!vol1
}
```

- Volume Quota service to monitor the volume quota status as shown below:

```
define service {
                 service_description            Volume Quota
- volume-name
                 host_name                      cluster-
name
                 use gluster-service-without-graph
                 _VOL_NAME                      volume-name
                 check_command
check_vol_quota_status!cluster-name!volume-name
                 notes                          Volume type
: Volume-Type
        }
```

- Volume Geo-Replication service to monitor Geo Replication status as shown below:

```
define service {
                 service_description            Volume Geo
Replication - volume-name
                 host_name                      cluster-
name
                 use gluster-service-without-graph
                 _VOL_NAME                      volume-name
                 check_command
check_vol_georep_status!cluster-name!volume-name
        }
```

3. In the **`/etc/nagios/gluster/`*`cluster-name`*** directory, create a file with name ***`host-name`*`.cfg`**. The host configuration for the node and service configuration for all the brick from the node are added in this file.

   In ***`host-name`*`.cfg`** file, add following definitions:

   a. Define Host for the node as shown below:

```
 define host {
         use                            gluster-host
         hostgroups    gluster_hosts,cluster-name
         alias                          host-name
         host_name                      host-name #Name given
by user to identify the node in Nagios
         _HOST_UUID                     host-uuid #Host UUID
returned by gluster peer status
         address                        host-address  # This
can be FQDN or IP address of the host
      }
```

   b. Create the following services for each brick in the node:

- Add *Brick Utilization* service as shown below:

```
define service {
                service_description            Brick
Utilization - brick-path
                 host_name                     host-name  #
Host name given in host definition
                 use                           brick-
service
                 _VOL_NAME                     Volume-Name
                 notes                          Volume :
Volume-Name
                 _BRICK_DIR                     brick-path
        }
```

- Add *Brick Status* service as shown below:

```
define service {
                 service_description            Brick -
brick-path
                 host_name                      host-name  #
Host name given in host definition
                 use           gluster-brick-status-service
                 _VOL_NAME                      Volume-Name
                 notes                          Volume :
Volume-Name
                 _BRICK_DIR                     brick-path
        }
```

4. Add host configurations and service configurations for all nodes in the cluster as shown in Step 3.

Configuring Red Hat Gluster Storage node

1. In **`/etc/nagios`** directory of each Red Hat Gluster Storage node, edit **`nagios_server.conf`** file by setting the configurations as shown below:

```
# NAGIOS SERVER
# The nagios server IP address or FQDN to which the NSCA command
# needs to be sent
[NAGIOS-SERVER]
nagios_server=NagiosServerIPAddress


# CLUSTER NAME
# The host name of the logical cluster configured in Nagios under
which
# the gluster volume services reside
[NAGIOS-DEFINTIONS]
cluster_name=cluster_auto


# LOCAL HOST NAME
# Host name given in the nagios server
[HOST-NAME]
hostname_in_nagios=NameOfTheHostInNagios

# LOCAL HOST CONFIGURATION
# Process monitoring sleeping intevel
[HOST-CONF]
proc-mon-sleep-time=TimeInSeconds
```

The **`nagios_server.conf`** file is used by **`glusterpmd`** service to get server name, host name, and the process monitoring interval time.

2. Start the **glusterpmd** service using the following command:

   ```
   # service glusterpmd start
   ```

Changing Nagios Monitoring time interval

By default, the active Red Hat Gluster Storage services are monitored every 10 minutes. You can change the time interval for monitoring by editing the **gluster-templates.cfg** file.

1. In **/etc/nagios/gluster/gluster-templates.cfg** file, edit the service with **gluster-service** name.

2. Add **normal_check_interval** and set the time interval to 1 to check all Red Hat Gluster Storage services every 1 minute as shown below:

   ```
   define service {
      name                         gluster-service
      use                          generic-service
      notifications_enabled        1
      notification_period          24x7
      notification_options         w,u,c,r,f,s
      notification_interval        120
      register                     0
      contacts                     +ovirt,snmp
      _GLUSTER_ENTITY              HOST_SERVICE
      normal_check_interval        1
   }
   ```

3. To change this on individual service, add this property to the required service definition as shown below:

   ```
   define service {
      name                    gluster-brick-status-service
      use                     gluster-service
      register                0
      event_handler           brick_status_event_handler
      check_command           check_brick_status
      normal_check_interval   1
   }
   ```

   The **check_interval** is controlled by the global directive **interval_length**. This defaults to 60 seconds. This can be changed in **/etc/nagios/nagios.cfg** as shown below:

   ```
   # INTERVAL LENGTH
   # This is the seconds per unit interval as used in the
   # host/contact/service configuration files.  Setting this to 60 means
   # that each interval is one minute long (60 seconds).  Other settings
   # have not been tested much, so your mileage is likely to vary...

   interval_length=TimeInSeconds
   ```

18.7. Troubleshooting Nagios

18.7.1. Troubleshooting NSCA and NRPE Configuration Issues

The possible errors while configuring Nagios Service Check Acceptor (NSCA) and Nagios Remote Plug-in Executor (NRPE) and the troubleshooting steps are listed in this section.

Troubleshooting NSCA Configuration Issues

- **Check Firewall and Port Settings on Nagios Server**

  If port 5667 is not opened on the server host's firewall, a timeout error is displayed. Ensure that port *5667* is opened.

  On Red Hat Gluster Storage based on Red Hat Enterprise Linux 6

  - Log in as root and run the following command on the Red Hat Gluster Storage node to get the list of current iptables rules:

    ```
    # iptables -L
    ```

  - The output is displayed as shown below:

    ```
    ACCEPT     tcp  --  anywhere             anywhere            tcp
    dpt:5667
    ```

  On Red Hat Gluster Storage based on Red Hat Enterprise Linux 7:

  - Run the following command on the Red Hat Gluster Storage node as root to get a listing of the current firewall rules:

    ```
    # firewall-cmd --list-all-zones
    ```

  - If the port is open, **`5667/tcp`** is listed beside **`ports:`** under one or more zones in your output.

- If the port is not open, add a firewall rule for the port:

  On Red Hat Gluster Storage based on Red Hat Enterprise Linux 6

  - If the port is not open, add an iptables rule by adding the following line in **`/etc/sysconfig/iptables`** file:

    ```
    -A INPUT -m state --state NEW -m tcp -p tcp --dport 5667 -j ACCEPT
    ```

  - Restart the iptables service using the following command:

    ```
    # service iptables restart
    ```

  - Restart the NSCA service using the following command:

    ```
    # service nsca restart
    ```

  On Red Hat Gluster Storage based on Red Hat Enterprise Linux 7:

  - Run the following commands to open the port:

```
# firewall-cmd --zone=zone_name --add-port=5667/tcp
# firewall-cmd --zone=zone_name --add-port=5667/tcp --permanent
```

- **Check the Configuration File on Red Hat Gluster Storage Node**

  Messages cannot be sent to the NSCA server, if Nagios server IP or FQDN, cluster name and hostname (as configured in Nagios server) are not configured correctly.

  Open the Nagios server configuration file */etc/nagios/nagios_server.conf* and verify if the correct configurations are set as shown below:

  ```
  # NAGIOS SERVER
  # The nagios server IP address or FQDN to which the NSCA command
  # needs to be sent
  [NAGIOS-SERVER]
  nagios_server=NagiosServerIPAddress


  # CLUSTER NAME
  # The host name of the logical cluster configured in Nagios under which
  # the gluster volume services reside
  [NAGIOS-DEFINTIONS]
  cluster_name=cluster_auto


  # LOCAL HOST NAME
  # Host name given in the nagios server
  [HOST-NAME]
  hostname_in_nagios=NagiosServerHostName
  ```

  If Host name is updated, restart the NSCA service using the following command:

  ```
  # service nsca restart
  ```

Troubleshooting NRPE Configuration Issues

- **CHECK_NRPE: Error - Could Not Complete SSL Handshake**

  This error occurs if the IP address of the Nagios server is not defined in the **`nrpe.cfg`** file of the Red Hat Gluster Storage node. To fix this issue, follow the steps given below:

  - Add the Nagios server IP address in **`/etc/nagios/nrpe.cfg`** file in the **`allowed_hosts`** line as shown below:

    ```
    allowed_hosts=127.0.0.1, NagiosServerIP
    ```

    The **`allowed_hosts`** is the list of IP addresses which can execute NRPE commands.

  - Save the **`nrpe.cfg`** file and restart NRPE service using the following command:

    ```
    # service nrpe restart
    ```

- **CHECK_NRPE: Socket Timeout After n Seconds**

  To resolve this issue perform the steps given below:

On Nagios Server:

The default timeout value for the NRPE calls is 10 seconds and if the server does not respond within 10 seconds, Nagios Server GUI displays an error that the NRPE call has timed out in 10 seconds. To fix this issue, change the timeout value for NRPE calls by modifying the command definition configuration files.

- Changing the NRPE timeout for services which directly invoke *check_nrpe*.

  For the services which directly invoke *check_nrpe* (check_disk_and_inode, check_cpu_multicore, and check_memory), modify the command definition configuration file **`/etc/nagios/gluster/gluster-commands.cfg`** by adding *-t Time in Seconds* as shown below:

  ```
  define command {
         command_name check_disk_and_inode
         command_line $USER1$/check_nrpe -H $HOSTADDRESS$ -c
  check_disk_and_inode -t TimeInSeconds
  }
  ```

- Changing the NRPE timeout for the services in **`nagios-server-addons`** package which invoke NRPE call through code.

  The services which invoke **`/usr/lib64/nagios/plugins/gluster/check_vol_server.py`** (check_vol_utilization, check_vol_status, check_vol_quota_status, check_vol_heal_status, and check_vol_georep_status) make NRPE call to the Red Hat Gluster Storage nodes for the details through code. To change the timeout for the NRPE calls, modify the command definition configuration file **`/etc/nagios/gluster/gluster-commands.cfg`** by adding *-t No of seconds* as shown below:

  ```
  define command {
        command_name check_vol_utilization
        command_line $USER1$/gluster/check_vol_server.py $ARG1$ $ARG2$
  -w $ARG3$ -c $ARG4$ -o utilization -t TimeInSeconds
  }
  ```

  The auto configuration service **`gluster_auto_discovery`** makes NRPE calls for the configuration details from the Red Hat Gluster Storage nodes. To change the NRPE timeout value for the auto configuration service, modify the command definition configuration file **`/etc/nagios/gluster/gluster-commands.cfg`** by adding *-t TimeInSeconds* as shown below:

  ```
  define command{
          command_name    gluster_auto_discovery
          command_line    sudo $USER1$/gluster/configure-gluster-
  nagios.py -H $ARG1$ -c $HOSTNAME$ -m auto -n $ARG2$ -t TimeInSeconds
  }
  ```

- Restart Nagios service using the following command:

  ```
   #  service nagios restart
  ```

On Red Hat Gluster Storage node:

- Add the Nagios server IP address as described in *CHECK_NRPE: Error - Could Not Complete SSL Handshake* section in *Troubleshooting NRPE Configuration Issues* section.

  - Edit the **nrpe.cfg** file using the following command:

    ```
    # vi /etc/nagios/nrpe.cfg
    ```

  - Search for the **command_timeout** and **connection_timeout** settings and change the value. The **command_timeout** value must be greater than or equal to the timeout value set in Nagios server.

    The timeout on checks can be set as *connection_timeout=300* and the *command_timeout=60* seconds.

  - Restart the NRPE service using the following command:

    ```
     #  service nrpe restart
    ```

- **Check the NRPE Service Status**

  This error occurs if the NRPE service is not running. To resolve this issue perform the steps given below:

  - Log in as root to the Red Hat Gluster Storage node and run the following command to verify the status of NRPE service:

    ```
    # service nrpe status
    ```

  - If NRPE is not running, start the service using the following command:

    ```
    # service nrpe start
    ```

- **Check Firewall and Port Settings**

  This error is associated with firewalls and ports. The timeout error is displayed if the NRPE traffic is not traversing a firewall, or if port *5666* is not open on the Red Hat Gluster Storage node.

  Ensure that port 5666 is open on the Red Hat Gluster Storage node.

  - Run **check_nrpe** command from the Nagios server to verify if the port is open and if NRPE is running on the Red Hat Gluster Storage Node .

  - Log into the Nagios server as root and run the following command:

    ```
    # /usr/lib64/nagios/plugins/check_nrpe -H RedHatStorageNodeIP
    ```

  - The output is displayed as given below:

    ```
    NRPE v2.14
    ```

  If not, ensure the that port 5666 is opened on the Red Hat Gluster Storage node.

  On Red Hat Gluster Storage based on Red Hat Enterprise Linux 6:

  - Run the following command on the Red Hat Gluster Storage node as root to get a listing of the current iptables rules:

    ```
    # iptables -L
    ```

  - If the port is open, the following appears in your output.

    ```
    ACCEPT     tcp  --  anywhere             anywhere            tcp
    dpt:5666
    ```

  On Red Hat Gluster Storage based on Red Hat Enterprise Linux 7:

  - Run the following command on the Red Hat Gluster Storage node as root to get a listing of the current firewall rules:

    ```
    # firewall-cmd --list-all-zones
    ```

  - If the port is open, **`5666/tcp`** is listed beside **`ports:`** under one or more zones in your output.

- If the port is not open, add an iptables rule for the port.

  On Red Hat Gluster Storage based on Red Hat Enterprise Linux 6:

  - To add iptables rule, edit the **`iptables`** file as shown below:

    ```
    # vi /etc/sysconfig/iptables
    ```

  - Add the following line in the file:

    ```
    -A INPUT -m state --state NEW -m tcp -p tcp --dport 5666 -j ACCEPT
    ```

  - Restart the iptables service using the following command:

    ```
    # service iptables restart
    ```

  - Save the file and restart the NRPE service:

    ```
    # service nrpe restart
    ```

  On Red Hat Gluster Storage based on Red Hat Enterprise Linux 7:

  - Run the following commands to open the port:

    ```
    # firewall-cmd --zone=zone_name --add-port=5666/tcp
    # firewall-cmd --zone=zone_name --add-port=5666/tcp --permanent
    ```

- **Checking Port 5666 From the Nagios Server with Telnet**

  Use telnet to verify the Red Hat Gluster Storage node's ports. To verify the ports of the Red Hat Gluster Storage node, perform the steps given below:

  - Log in as root on Nagios server.

  - Test the connection on port 5666 from the Nagios server to the Red Hat Gluster Storage node using the following command:

    ```
    # telnet RedHatStorageNodeIP 5666
    ```

  - The output displayed is similar to:

```
telnet 10.70.36.49 5666
Trying 10.70.36.49...
Connected to 10.70.36.49.
Escape character is '^]'.
```

- **Connection Refused By Host**

  This error is due to port/firewall issues or incorrectly configured *allowed_hosts* directives. See the sections *CHECK_NRPE: Error - Could Not Complete SSL Handshake* and *CHECK_NRPE: Socket Timeout After n Seconds* for troubleshooting steps.

Chapter 19. Monitoring Red Hat Gluster Storage Gluster Workload

Monitoring storage volumes is helpful when conducting a capacity planning or performance tuning activity on a Red Hat Gluster Storage volume. You can monitor the Red Hat Gluster Storage volumes with different parameters and use those system outputs to identify and troubleshoot issues.

You can use the **`volume top`** and **`volume profile`** commands to view vital performance information and identify bottlenecks on each brick of a volume.

You can also perform a statedump of the brick processes and NFS server process of a volume, and also view volume status and volume information.

> **Note**
>
> If you restart the server process, the existing **`profile`** and **`top`** information will be reset.

19.1. Profiling volumes

19.1.1. Server-side volume profiling using `volume profile`

The **`volume profile`** command provides an interface to get the per-brick or NFS server I/O information for each File Operation (FOP) of a volume. This information helps in identifying the bottlenecks in the storage system.

This section describes how to use the **`volume profile`** command.

19.1.1.1. Start Profiling

To view the file operation information of each brick, start the profiling command:

```
# gluster volume profile VOLNAME start
```

For example, to start profiling on *test-volume*:

```
# gluster volume profile test-volume start
Profiling started on test-volume
```

> **Important**
>
> Running **`profile`** command can affect system performance while the profile information is being collected. Red Hat recommends that profiling should only be used for debugging.

When profiling is started on the volume, the following additional options are displayed when using the **`volume info`** command:

```
diagnostics.count-fop-hits: on
diagnostics.latency-measurement: on
```

19.1.1.2. Displaying the I/O Information

To view the I/O information of the bricks on a volume, use the following command:

```
# gluster volume profile VOLNAME info
```

For example, to view the I/O information of *test-volume*:

```
# gluster volume profile test-volume info
Brick: Server1:/rhgs/brick0/2
Cumulative Stats:

Block                     1b+           32b+           64b+
Size:
       Read:                0              0              0
       Write:             908             28              8

Block                   128b+           256b+         512b+
Size:
       Read:                0               6             4
       Write:               5              23            16

Block                  1024b+          2048b+        4096b+
Size:
       Read:                 0              52           17
       Write:               15             120          846

Block                   8192b+         16384b+      32768b+
Size:
       Read:                52               8           34
       Write:              234             134          286

Block                                  65536b+     131072b+
Size:
       Read:                               118          622
       Write:                             1341          594


%-latency  Avg-      Min-       Max-       calls     Fop
          latency   Latency    Latency

___________________________________________________________
4.82      1132.28   21.00      800970.00   4575    WRITE
5.70       156.47    9.00      665085.00   39163   READDIRP
11.35      315.02    9.00     1433947.00   38698   LOOKUP
11.88     1729.34   21.00     2569638.00    7382   FXATTROP
47.35   104235.02 2485.00     7789367.00     488   FSYNC

------------------

------------------

Duration     : 335

BytesRead    : 94505058

BytesWritten : 195571980
```

To view the I/O information of the NFS server on a specified volume, use the following command:

```
# gluster volume profile VOLNAME info nfs
```

For example, to view the I/O information of the NFS server on *test-volume*:

```
# gluster volume profile test-volume info nfs
NFS Server : localhost
----------------------
Cumulative Stats:
Block Size:              32768b+               65536b+
No. of Reads:                    0                     0
No. of Writes:                 1000                  1000
%-latency   Avg-latency   Min-Latency   Max-Latency   No. of calls
Fop
---------   -----------   -----------   -----------   ------------
----
0.01     410.33 us     194.00 us     641.00 us              3      STATFS
0.60     465.44 us     346.00 us     867.00 us            147       FSTAT
1.63     187.21 us      67.00 us    6081.00 us           1000     SETATTR
1.94     221.40 us      58.00 us   55399.00 us           1002      ACCESS
2.55     301.39 us      52.00 us   75922.00 us            968        STAT
2.85     326.18 us      88.00 us   66184.00 us           1000    TRUNCATE
4.47     511.89 us      60.00 us  101282.00 us           1000       FLUSH
5.02    3907.40 us    1723.00 us   19508.00 us            147    READDIRP
25.42    2876.37 us     101.00 us  843209.00 us           1012      LOOKUP
55.52    3179.16 us     124.00 us  121158.00 us           2000       WRITE

Duration: 7074 seconds
Data Read: 0 bytes
Data Written: 102400000 bytes

Interval 1 Stats:
Block Size:              32768b+               65536b+
No. of Reads:                    0                     0
No. of Writes:                 1000                  1000
%-latency   Avg-latency   Min-Latency   Max-Latency   No. of calls
Fop
---------   -----------   -----------   -----------   ------------
----
0.01     410.33 us     194.00 us     641.00 us              3      STATFS
0.60     465.44 us     346.00 us     867.00 us            147       FSTAT
1.63     187.21 us      67.00 us    6081.00 us           1000     SETATTR
1.94     221.40 us      58.00 us   55399.00 us           1002      ACCESS
2.55     301.39 us      52.00 us   75922.00 us            968        STAT
2.85     326.18 us      88.00 us   66184.00 us           1000    TRUNCATE
4.47     511.89 us      60.00 us  101282.00 us           1000       FLUSH
5.02    3907.40 us    1723.00 us   19508.00 us            147    READDIRP
25.41    2878.07 us     101.00 us  843209.00 us           1011      LOOKUP
55.53    3179.16 us     124.00 us  121158.00 us           2000       WRITE

Duration: 330 seconds
Data Read: 0 bytes
Data Written: 102400000 bytes
```

19.1.1.3. Stop Profiling

To stop profiling on a volume, use the following command:

# gluster volume profile *VOLNAME* stop

For example, to stop profiling on *test-volume*:

```
# gluster volume profile test-volume stop
    Profiling stopped on test-volume
```

19.1.2. Client-side volume profiling (FUSE only)

Red Hat Gluster Storage lets you profile how your mount point is being accessed, so that you can investigate latency issues even when you cannot instrument the application accessing your storage.

The io-stats translator records statistics of all file system activity on a Red Hat Gluster Storage volume that travels through a FUSE mount point. It collects information on files opened from the FUSE mount path, the read and write throughput for these files, the number of blocks read and written, and the latency observed for different file operations.

Run the following command to output all recorded statistics for the specified mount point to the specified output file.

```
# setfattr -n trusted.io-stats-dump -v output_file_id mount_point
```

This generates a number of files in the **/var/run/gluster** directory. The *output_file_id* is not the whole file name, but is used as part of the name of the generated files.

19.2. Running the Volume Top Command

The **volume top** command allows you to view the glusterFS bricks’ performance metrics, including read, write, file open calls, file read calls, file write calls, directory open calls, and directory real calls. The **volume top** command displays up to 100 results.

This section describes how to use the **volume top** command.

19.2.1. Viewing Open File Descriptor Count and Maximum File Descriptor Count

You can view the current open file descriptor count and the list of files that are currently being accessed on the brick with the **volume top** command. The **volume top** command also displays the maximum open file descriptor count of files that are currently open, and the maximum number of files opened at any given point of time since the servers are up and running. If the brick name is not specified, then the open file descriptor metrics of all the bricks belonging to the volume displays.

To view the open file descriptor count and the maximum file descriptor count, use the following command:

# gluster volume top *VOLNAME* open [nfs | brick *BRICK-NAME*] [list-cnt *cnt*]

For example, to view the open file descriptor count and the maximum file descriptor count on brick *server:/export* on *test-volume*, and list the top 10 open calls:

```
# gluster volume top test-volume open brick server:/export  list-cnt 10
Brick: Server1:/rhgs/brick1
Current open fd's: 34 Max open fd's: 209
             ==========Open file stats========
open                file name
call count

2                   /clients/client0/~dmtmp/PARADOX/
                    COURSES.DB

11                  /clients/client0/~dmtmp/PARADOX/
                    ENROLL.DB

11                  /clients/client0/~dmtmp/PARADOX/
                    STUDENTS.DB

10                  /clients/client0/~dmtmp/PWRPNT/
                    TIPS.PPT

10                  /clients/client0/~dmtmp/PWRPNT/
                    PCBENCHM.PPT

9                   /clients/client7/~dmtmp/PARADOX/
                    STUDENTS.DB

9                   /clients/client1/~dmtmp/PARADOX/
                    STUDENTS.DB

9                   /clients/client2/~dmtmp/PARADOX/
                    STUDENTS.DB

9                   /clients/client0/~dmtmp/PARADOX/
                    STUDENTS.DB

9                   /clients/client8/~dmtmp/PARADOX/
                    STUDENTS.DB
```

19.2.2. Viewing Highest File Read Calls

You can view a list of files with the highest file read calls on each brick with the **volume top** command. If the brick name is not specified, a list of 100 files are displayed by default.

To view the highest read() calls, use the following command:

```
# gluster volume top VOLNAME read [nfs | brick BRICK-NAME] [list-cnt cnt]
```

For example, to view the highest read calls on brick *server:/export* of *test-volume*:

```
# gluster volume top test-volume read brick server:/export list-cnt 10
Brick: server:/export/dir1
          ==========Read file stats========

read                filename
call count

116                /clients/client0/~dmtmp/SEED/LARGE.FIL

64                 /clients/client0/~dmtmp/SEED/MEDIUM.FIL

54                 /clients/client2/~dmtmp/SEED/LARGE.FIL
```

```
54               /clients/client6/~dmtmp/SEED/LARGE.FIL

54               /clients/client5/~dmtmp/SEED/LARGE.FIL

54               /clients/client0/~dmtmp/SEED/LARGE.FIL

54               /clients/client3/~dmtmp/SEED/LARGE.FIL

54               /clients/client4/~dmtmp/SEED/LARGE.FIL

54               /clients/client9/~dmtmp/SEED/LARGE.FIL

54               /clients/client8/~dmtmp/SEED/LARGE.FIL
```

19.2.3. Viewing Highest File Write Calls

You can view a list of files with the highest file write calls on each brick with the **volume top** command. If the brick name is not specified, a list of 100 files displays by default.

To view the highest write() calls, use the following command:

```
# gluster volume top VOLNAME write [nfs | brick BRICK-NAME] [list-cnt cnt]
```

For example, to view the highest write calls on brick *server:/export* of *test-volume*:

```
# gluster volume top test-volume write brick server:/export/ list-cnt 10
Brick: server:/export/dir1

               ==========Write file stats========
write call count   filename

83                /clients/client0/~dmtmp/SEED/LARGE.FIL

59                /clients/client7/~dmtmp/SEED/LARGE.FIL

59                /clients/client1/~dmtmp/SEED/LARGE.FIL

59                /clients/client2/~dmtmp/SEED/LARGE.FIL

59                /clients/client0/~dmtmp/SEED/LARGE.FIL

59                /clients/client8/~dmtmp/SEED/LARGE.FIL

59                /clients/client5/~dmtmp/SEED/LARGE.FIL

59                /clients/client4/~dmtmp/SEED/LARGE.FIL

59                /clients/client6/~dmtmp/SEED/LARGE.FIL

59                /clients/client3/~dmtmp/SEED/LARGE.FIL
```

19.2.4. Viewing Highest Open Calls on a Directory

You can view a list of files with the highest open calls on the directories of each brick with the **volume top** command. If the brick name is not specified, the metrics of all bricks belonging to that volume displays.

To view the highest open() calls on each directory, use the following command:

```
# gluster volume top VOLNAME opendir [brick BRICK-NAME] [list-cnt cnt]
```

For example, to view the highest open calls on brick *server:/export/* of *test-volume*:

```
# gluster volume top test-volume opendir brick server:/export/ list-cnt 10
Brick: server:/export/dir1
         ==========Directory open stats========

Opendir count     directory name

1001              /clients/client0/~dmtmp

454               /clients/client8/~dmtmp

454               /clients/client2/~dmtmp

454               /clients/client6/~dmtmp

454               /clients/client5/~dmtmp

454               /clients/client9/~dmtmp

443               /clients/client0/~dmtmp/PARADOX

408               /clients/client1/~dmtmp

408               /clients/client7/~dmtmp

402               /clients/client4/~dmtmp
```

19.2.5. Viewing Highest Read Calls on a Directory

You can view a list of files with the highest directory read calls on each brick with the **volume top** command. If the brick name is not specified, the metrics of all bricks belonging to that volume displays.

To view the highest directory read() calls on each brick, use the following command:

```
# gluster volume top VOLNAME readdir [nfs | brick BRICK-NAME] [list-cnt cnt]
```

For example, to view the highest directory read calls on brick *server:/export/* of *test-volume*:

```
# gluster volume top test-volume readdir brick server:/export/ list-cnt 10
Brick: server:/export/dir1
==========Directory readdirp stats========

readdirp count           directory name

1996                    /clients/client0/~dmtmp

1083                    /clients/client0/~dmtmp/PARADOX

```

```
904                     /clients/client8/~dmtmp

904                     /clients/client2/~dmtmp

904                     /clients/client6/~dmtmp

904                     /clients/client5/~dmtmp

904                     /clients/client9/~dmtmp

812                     /clients/client1/~dmtmp

812                     /clients/client7/~dmtmp

800                     /clients/client4/~dmtmp
```

19.2.6. Viewing Read Performance

You can view the read throughput of files on each brick with the **volume top** command. If the brick name is not specified, the metrics of all the bricks belonging to that volume is displayed. The output is the read throughput.

This command initiates a read() call for the specified count and block size and measures the corresponding throughput directly on the back-end export, bypassing glusterFS processes.

To view the read performance on each brick, use the command, specifying options as needed:

# gluster volume top *VOLNAME* read-perf [bs *blk-size* count *count*] [nfs | brick *BRICK-NAME*] [list-cnt *cnt*]

For example, to view the read performance on brick **server:/export/** of *test-volume*, specifying a 256 block size, and list the top 10 results:

```
# gluster volume top test-volume read-perf bs 256 count 1 brick
server:/export/ list-cnt 10
Brick: server:/export/dir1 256 bytes (256 B) copied, Throughput: 4.1 MB/s
       ==========Read throughput file stats========

read         filename                         Time
through
put(MBp
s)

2912.00   /clients/client0/~dmtmp/PWRPNT/    -2012-05-09
           TRIDOTS.POT                      15:38:36.896486

2570.00   /clients/client0/~dmtmp/PWRPNT/    -2012-05-09
           PCBENCHM.PPT                     15:38:39.815310

2383.00   /clients/client2/~dmtmp/SEED/      -2012-05-09
           MEDIUM.FIL                       15:52:53.631499

2340.00   /clients/client0/~dmtmp/SEED/      -2012-05-09
           MEDIUM.FIL                       15:38:36.926198

2299.00   /clients/client0/~dmtmp/SEED/      -2012-05-09
```

```
           LARGE.FIL                     15:38:36.930445

2259.00  /clients/client0/~dmtmp/PARADOX/    -2012-05-09
          COURSES.X04                    15:38:40.549919

2221.00  /clients/client9/~dmtmp/PARADOX/    -2012-05-09
          STUDENTS.VAL                   15:52:53.298766

2221.00  /clients/client8/~dmtmp/PARADOX/    -2012-05-09
         COURSES.DB                      15:39:11.776780

2184.00  /clients/client3/~dmtmp/SEED/       -2012-05-09
          MEDIUM.FIL                     15:39:10.251764

2184.00  /clients/client5/~dmtmp/WORD/       -2012-05-09
         BASEMACH.DOC                    15:39:09.336572
```

19.2.7. Viewing Write Performance

You can view the write throughput of files on each brick or NFS server with the **`volume top`** command. If brick name is not specified, then the metrics of all the bricks belonging to that volume will be displayed. The output will be the write throughput.

This command initiates a write operation for the specified count and block size and measures the corresponding throughput directly on back-end export, bypassing glusterFS processes.

To view the write performance on each brick, use the following command, specifying options as needed:

`# gluster volume top` ***`VOLNAME`*** **`write-perf [bs`** ***`blk-size`*** **`count`** ***`count`*** **`] [nfs | brick`** ***`BRICK-NAME`*** **`] [list-cnt`** ***`cnt`*** **`]`**

For example, to view the write performance on brick **`server:/export/`** of *test-volume*, specifying a 256 block size, and list the top 10 results:

```
# gluster volume top test-volume write-perf bs 256 count 1 brick
server:/export/ list-cnt 10
Brick: server:/export/dir1 256 bytes (256 B) copied, Throughput: 2.8 MB/s
       ==========Write throughput file stats========

write                filename                         Time
throughput
(MBps)

1170.00    /clients/client0/~dmtmp/SEED/     -2012-05-09
           SMALL.FIL                     15:39:09.171494

1008.00    /clients/client6/~dmtmp/SEED/     -2012-05-09
           LARGE.FIL                      15:39:09.73189

949.00    /clients/client0/~dmtmp/SEED/      -2012-05-09
          MEDIUM.FIL                     15:38:36.927426

936.00   /clients/client0/~dmtmp/SEED/       -2012-05-09
         LARGE.FIL                        15:38:36.933177
897.00   /clients/client5/~dmtmp/SEED/       -2012-05-09
         MEDIUM.FIL                       15:39:09.33628
```

```
897.00   /clients/client6/~dmtmp/SEED/       -2012-05-09
         MEDIUM.FIL                        15:39:09.27713

885.00   /clients/client0/~dmtmp/SEED/       -2012-05-09
          SMALL.FIL                       15:38:36.924271

528.00   /clients/client5/~dmtmp/SEED/       -2012-05-09
         LARGE.FIL                         15:39:09.81893

516.00   /clients/client6/~dmtmp/ACCESS/    -2012-05-09
         FASTENER.MDB                     15:39:01.797317
```

19.3. gstatus Command

19.3.1. gstatus Command

A Red Hat Gluster Storage trusted storage pool consists of nodes, volumes, and bricks. A new command called **gstatus** provides an overview of the health of a Red Hat Gluster Storage trusted storage pool for distributed, replicated, distributed-replicated, dispersed, and distributed-dispersed volumes.

The **gstatus** command provides an easy-to-use, high-level view of the health of a trusted storage pool with a single command. By executing the glusterFS commands, it gathers information about the statuses of the Red Hat Gluster Storage nodes, volumes, and bricks. The checks are performed across the trusted storage pool and the status is displayed. This data can be analyzed to add further checks and incorporate deployment best-practices and free-space triggers.

A Red Hat Gluster Storage volume is made from individual file systems (glusterFS bricks) across multiple nodes. Although the complexity is abstracted, the status of the individual bricks affects the data availability of the volume. For example, even without replication, the loss of a single brick in the volume will not cause the volume itself to be unavailable, instead this would manifest as inaccessible files in the file system.

Note

As of Rad Hat Gluster Storage 3.4, the gstatus command is deprecated. To view health of Red Hat Gluster Storage clusters and volumes, access the Grafana Dashboard integrated to the Web Administration environment.

19.3.1.1. Prerequisites

Package dependencies

- Python 2.6 or above

To install gstatus, refer to the *Deploying gstatus on Red Hat Gluster Storage* chapter in the *Red Hat Gluster Storage 3.4 Installation Guide*.

19.3.2. Executing the gstatus command

The **gstatus** command can be invoked in different ways. The table below shows the optional switches that can be used with gstatus.

```
# gstatus -h
Usage: gstatus [options]
```

Table 19.1. gstatus Command Options

Option	Description
--version	Displays the program's version number and exits.
-h, --help	Displays the help message and exits.
-s, --state	Displays the high level health of the Red Hat Gluster Storage trusted storage pool.
-v, --volume	Displays volume information of all the volumes, by default. Specify a volume name to display the volume information of a specific volume.
-b, --backlog	Probes the self heal state.
-a, --all	Displays the detailed status of volume health. (This output is aggregation of -s and -v).
-l, --layout	Displays the brick layout when used in combination with -v, or -a .
-o OUTPUT_MODE, --output-mode=OUTPUT_MODE	Produces outputs in various formats such as - json, keyvalue, or console(default).
-D, --debug	Enables the debug mode.
-w, --without-progress	Disables progress updates during data gathering.
-u UNITS, --units=UNITS	Displays capacity units in decimal or binary format (GB vs GiB).
-t TIMEOUT, --timeout=TIMEOUT	Specify the command timeout value in seconds.

Table 19.2. Commonly used gstatus Commands

Command	Description
`gstatus -s`	An overview of the trusted storage pool.
`gstatus -a`	View detailed status of the volume health.
`gstatus -vl` ***`VOLNAME`***	View the volume details, including the brick layout.
`gstatus -o` ***`<keyvalue>`***	View the summary output for Nagios and Logstash.

Interpreting the output with Examples

Each invocation of **`gstatus`** provides a header section, which provides a high level view of the state of the Red Hat Gluster Storage trusted storage pool. The **`Status`** field within the header offers two states; **`Healthy`** and **`Unhealthy`**. When problems are detected, the status field changes to Unhealthy(n), where n denotes the total number of issues that have been detected.

The following examples illustrate **`gstatus`** command output for both healthy and unhealthy Red Hat Gluster Storage environments.

Example 19.1. Example 1: Trusted Storage Pool is in a healthy state; all nodes, volumes and bricks are online

```
# gstatus -a

   Product: RHGS Server v3.2.0      Capacity:  36.00 GiB(raw bricks)
      Status: HEALTHY                          7.00 GiB(raw used)
   Glusterfs: 3.7.1                           18.00 GiB(usable from volumes)
```

```
  OverCommit: No                Snapshots:   0

   Nodes    :  4/ 4  Volumes:  1 Up
   Self Heal:  4/ 4           0 Up(Degraded)
   Bricks   :  4/ 4           0 Up(Partial)
   Connections  : 5 / 20       0 Down

Volume Information
 splunk       UP - 4/4 bricks up - Distributed-Replicate
                  Capacity: (18% used) 3.00 GiB/18.00 GiB (used/total)
                  Snapshots: 0
                  Self Heal:  4/ 4
                  Tasks Active: None
                  Protocols: glusterfs:on  NFS:on  SMB:off
                  Gluster Connectivty: 5 hosts, 20 tcp connections



Status Messages
- Cluster is HEALTHY, all_bricks checks successful
```

Example 19.2. Example 2: A node is down within the trusted pool

```
# gstatus -al

     Product: RHGS Server v3.1.1      Capacity:  27.00 GiB(raw bricks)
      Status: UNHEALTHY(4)                   5.00 GiB(raw used)
   Glusterfs: 3.7.1                      18.00 GiB(usable from volumes)
  OverCommit: No                Snapshots:   0

   Nodes    :  3/ 4  Volumes:  0 Up
   Self Heal:  3/ 4           1 Up(Degraded)
   Bricks   :  3/ 4           0 Up(Partial)
   Connections  :  5/ 20       0 Down

Volume Information
 splunk           UP(DEGRADED) - 3/4 bricks up - Distributed-Replicate
                  Capacity: (18% used) 3.00 GiB/18.00 GiB (used/total)
                  Snapshots: 0
                  Self Heal:  3/ 4
                  Tasks Active: None
                  Protocols: glusterfs:on  NFS:on  SMB:off
                  Gluster Connectivty: 5 hosts, 20 tcp connections

 splunk---------- +
                  |
                Distribute (dht)
                         |
                         +-- Repl Set 0 (afr)
                         |     |
                         |     +--splunk-rhs1:/rhgs/brick1/splunk(UP)
2.00 GiB/9.00 GiB
                         |     |
                         |     +--splunk-rhs2:/rhgs/brick1/splunk(UP)
2.00 GiB/9.00 GiB
```

```
                         |
                         +-- Repl Set 1 (afr)
                               |
                               +--splunk-rhs3:/rhgs/brick1/splunk(DOWN)
0.00 KiB/0.00 KiB
                               |
                               +--splunk-rhs4:/rhgs/brick1/splunk(UP)
2.00 GiB/9.00 GiB
 Status Messages
  - Cluster is UNHEALTHY
  - One of the nodes in the cluster is down
  - Brick splunk-rhs3:/rhgs/brick1/splunk in volume 'splunk' is
down/unavailable
  - INFO -> Not all bricks are online, so capacity provided is NOT accurate
```

Example 2, displays the output of the command when the **`-l`** option is used. The **`brick layout`** mode shows the brick and node relationships. This provides a simple means of checking the replication relationships for bricks across nodes is as intended.

Table 19.3. Field Descriptions of the `gstatus` command output

Field	Description
Volume State	Up – The volume is started and available, and all the bricks are up .
	Up (Degraded) - This state is specific to replicated volumes, where at least one brick is down within a replica set. Data is still 100% available due to the alternate replicas, but the resilience of the volume to further failures within the same replica set flags this volume as **`degraded`**.
	Up (Partial) - Effectively, this means that all though some bricks in the volume are online, there are others that are down to a point where areas of the file system will be missing. For a distributed volume, this state is seen if any brick is down, whereas for a replicated volume a complete replica set needs to be down before the volume state transitions to **`PARTIAL`**.
	Down - Bricks are down, or the volume is yet to be started.
Capacity Information	This information is derived from the brick information taken from the **`volume status detail`** command. The accuracy of this number hence depends on the nodes and bricks all being online - elements missing from the configuration are not considered in the calculation.
Over-commit Status	The physical file system used by a brick could be re-used by multiple volumes, this field indicates whether a brick is used by multiple volumes. But this exposes the system to capacity conflicts across different volumes when the quota feature is not in use. Reusing a brick for multiple volumes is not recommended.

Field	Description
Connections	Displays a count of connections made to the trusted pool and each of the volumes.
Nodes / Self Heal / Bricks X/Y	This indicates that X components of Y total/expected components within the trusted pool are online. In Example 2, note that 3/4 is displayed against all of these fields, indicating 3 nodes are available out of 4 nodes. A node, brick, and the self-heal daemon are also unavailable.
Tasks Active	Active background tasks such as rebalance, remove-brick are displayed here against individual volumes.
Protocols	Displays which protocols have been enabled for the volume.
Snapshots	Displays a count of the number of snapshots taken for the volume. The snapshot count for each volume is **`rolled up`** to the trusted storage pool to provide a high level view of the number of snapshots in the environment.
Status Messages	After the information is gathered, any errors detected are reported in the **`Status Messages`** section. These descriptions provide a view of the problem and the potential impact of the condition.

19.4. Listing Volumes

You can list all volumes in the trusted storage pool using the following command:

```
# gluster volume list
```

For example, to list all volumes in the trusted storage pool:

```
# gluster volume list
test-volume
volume1
volume2
volume3
```

19.5. Displaying Volume Information

You can display information about a specific volume, or all volumes, as needed, using the following command:

```
# gluster volume info VOLNAME
```

For example, to display information about *test-volume*:

```
# gluster volume info test-volume
Volume Name: test-volume
Type: Distribute
Status: Created
Number of Bricks: 4
Bricks:
```

```
Brick1: server1:/rhgs/brick1
Brick2: server2:/rhgs/brick2
Brick3: server3:/rhgs/brick3
Brick4: server4:/rhgs/brick4
```

19.6. Obtaining Node Information

A Red Hat Gluster Storage trusted storage pool consists of nodes, volumes, and bricks. The **get-state** command outputs information about a node to a specified file.

Using the command line interface, external applications can invoke the command on all nodes of the trusted storage pool, and parse and collate the data obtained from all these nodes to get an easy-to-use and complete picture of the state of the trusted storage pool in a machine parseable format.

Executing the get-state Command

The **get-state** command outputs information about a node to a specified file and can be invoked in different ways. The table below shows the options that can be used with the get-state command.

```
#  gluster get-state [odir path_to_output_dir] [file filename]
[detail|volumeoptions]
Usage:  get-state [options]
```

Table 19.4. get-state Command Options

Command	Description
gluster get-state	**glusterd** state information is saved in the **/var/run/gluster/glusterd_state_*timestamp*** file.
gluster get-state file *filename*	**glusterd** state information is saved in the **/var/run/gluster/** directory with the *filename* as specified in the command.
gluster get-state odir *directory* file *filename*	**glusterd** state information is saved in the directory and in the file name as specified in the command.
gluster get-state detail	**glusterd** state information is saved in the **/var/run/gluster/glusterd_state_*timestamp*** file, and all clients connected per brick are included in the output.
gluster get-state volumeoptions	**glusterd** state information is saved in the **/var/run/gluster/glusterd_state_*timestamp*** file, and all values for all the volume options are included in the output.

Interpreting the Output with Examples

Invocation of the **get-state** command saves the information that reflects the node level status of the trusted storage pool as maintained in glusterd (no other daemons are supported as of now) to a file specified in the command. By default, the output will be dumped to **/var/run/gluster/glusterd_state_*timestamp*** file .

Invocation of the get-state command provides the following information:

Table 19.5. Output Description

Section	Description
Global	Displays the UUID and the op-version of the glusterd.
Global options	Displays cluster specific options that have been set explicitly through the volume set command.
Peers	Displays the peer node information including its hostname and connection status.
Volumes	Displays the list of volumes created on this node along with the detailed information on each volume.
Services	Displays the list of the services configured on this node along with its status.
Misc	Displays miscellaneous information about the node. For example, configured ports.

Example Output for `gluster get-state`:

```
# gluster get-state
glusterd state dumped to /var/run/gluster/glusterd_state_timestamp
```

View the file using the **`cat`** ***`state_dump_file_path`*** command:

```
[Global]
MYUUID: 1e20ed87-c22a-4612-ab04-90765bccaea5
op-version: 31305


[Global options]
cluster.server-quorum-ratio: 60


[Peers]
Peer1.primary_hostname: output omitted
Peer1.uuid: dfc7ff96-b61d-4c88-a3ad-b6852f72c5f0
Peer1.state: Peer in Cluster
Peer1.connected: Connected
Peer1.othernames:
Peer2.primary_hostname: output omitted
Peer2.uuid: dd83409e-22fa-4186-935a-648a1927cc9d
Peer2.state: Peer in Cluster
Peer2.connected: Connected
Peer2.othernames:


[Volumes]
Volume1.name: tv1
Volume1.id: cf89d345-8cde-4c53-be85-1f3f20e7e410
Volume1.type: Distribute
Volume1.transport_type: tcp
Volume1.status: Started
Volume1.brickcount: 3
Volume1.Brick1.path: output omitted:/root/bricks/tb11
Volume1.Brick1.hostname: output omitted
Volume1.Brick1.port: 49152
Volume1.Brick1.rdma_port: 0
```

```
Volume1.Brick1.status: Started
Volume1.Brick1.signedin: True
Volume1.Brick2.path: output omitted:/root/bricks/tb12
Volume1.Brick2.hostname: output omitted
Volume1.Brick3.path: output omitted:/root/bricks/tb13
Volume1.Brick3.hostname: output omitted
Volume1.snap_count: 0
Volume1.stripe_count: 1
Volume1.replica_count: 1
Volume1.subvol_count: 3
Volume1.arbiter_count: 0
Volume1.disperse_count: 0
Volume1.redundancy_count: 0
Volume1.quorum_status: not_applicable
Volume1.snapd_svc.online_status: Online
Volume1.snapd_svc.inited: True
Volume1.rebalance.id: 00000000-0000-0000-0000-000000000000
Volume1.rebalance.status: not_started
Volume1.rebalance.failures: 0
Volume1.rebalance.skipped: 0
Volume1.rebalance.lookedup: 0
Volume1.rebalance.files: 0
Volume1.rebalance.data: 0Bytes
[Volume1.options]
features.uss: on
transport.address-family: inet
performance.readdir-ahead: on
nfs.disable: on


Volume2.name: tv2
Volume2.id: 700fd588-6fc2-46d5-9435-39c434656fe2
Volume2.type: Distribute
Volume2.transport_type: tcp
Volume2.status: Created
Volume2.brickcount: 3
Volume2.Brick1.path: output omitted:/root/bricks/tb21
Volume2.Brick1.hostname: output omitted
Volume2.Brick1.port: 0
Volume2.Brick1.rdma_port: 0
Volume2.Brick1.status: Stopped
Volume2.Brick1.signedin: False
Volume2.Brick2.path: output omitted:/root/bricks/tb22
Volume2.Brick2.hostname: output omitted
Volume2.Brick3.path: output omitted:/root/bricks/tb23
Volume2.Brick3.hostname: output omitted
Volume2.snap_count: 0
Volume2.stripe_count: 1
Volume2.replica_count: 1
Volume2.subvol_count: 3
Volume2.arbiter_count: 0
Volume2.disperse_count: 0
Volume2.redundancy_count: 0
Volume2.quorum_status: not_applicable
Volume2.snapd_svc.online_status: Offline
Volume2.snapd_svc.inited: False
```

```
Volume2.rebalance.id: 00000000-0000-0000-0000-000000000000
Volume2.rebalance.status: not_started
Volume2.rebalance.failures: 0
Volume2.rebalance.skipped: 0
Volume2.rebalance.lookedup: 0
Volume2.rebalance.files: 0
Volume2.rebalance.data: 0Bytes
[Volume2.options]
transport.address-family: inet
performance.readdir-ahead: on
nfs.disable: on


Volume3.name: tv3
Volume3.id: 97b94d77-116a-4595-acfc-9676e4ebcbd2
Volume3.type: Tier
Volume3.transport_type: tcp
Volume3.status: Stopped
Volume3.brickcount: 4
Volume3.Brick1.path: output omitted:/root/bricks/tb34
Volume3.Brick1.hostname: output omitted
Volume3.Brick2.path: output omitted:/root/bricks/tb33
Volume3.Brick2.hostname: output omitted
Volume3.Brick3.path: output omitted:/root/bricks/tb31
Volume3.Brick3.hostname: output omitted
Volume3.Brick3.port: 49154
Volume3.Brick3.rdma_port: 0
Volume3.Brick3.status: Stopped
Volume3.Brick3.signedin: False
Volume3.Brick3.tier: Cold
Volume3.Brick4.path: output omitted:/root/bricks/tb32
Volume3.Brick4.hostname: output omitted
Volume3.snap_count: 0
Volume3.stripe_count: 1
Volume3.replica_count: 2
Volume3.subvol_count: 2
Volume3.arbiter_count: 0
Volume3.disperse_count: 0
Volume3.redundancy_count: 0
Volume3.quorum_status: not_applicable
Volume3.snapd_svc.online_status: Offline
Volume3.snapd_svc.inited: True
Volume3.rebalance.id: 00000000-0000-0000-0000-000000000000
Volume3.rebalance.status: not_started
Volume3.rebalance.failures: 0
Volume3.rebalance.skipped: 0
Volume3.rebalance.lookedup: 0
Volume3.rebalance.files: 0
Volume3.rebalance.data: 0Bytes
Volume3.tier_info.cold_tier_type: Replicate
Volume3.tier_info.cold_brick_count: 2
Volume3.tier_info.cold_replica_count: 2
Volume3.tier_info.cold_disperse_count: 0
Volume3.tier_info.cold_dist_leaf_count: 2
Volume3.tier_info.cold_redundancy_count: 0
Volume3.tier_info.hot_tier_type: Replicate
```

```
Volume3.tier_info.hot_brick_count: 2
Volume3.tier_info.hot_replica_count: 2
Volume3.tier_info.promoted: 0
Volume3.tier_info.demoted: 0
[Volume3.options]
cluster.tier-mode: cache
features.ctr-enabled: on
transport.address-family: inet
performance.readdir-ahead: on
nfs.disable: on


Volume4.name: tv4
Volume4.id: ad7260ac-0d5c-461f-a39c-a0f4a4ff854b
Volume4.type: Distribute
Volume4.transport_type: tcp
Volume4.status: Started
Volume4.brickcount: 2
Volume4.Brick1.path: output omitted:/root/bricks/tb41
Volume4.Brick1.hostname: output omitted
Volume4.Brick2.path: output omitted:/root/bricks/tb42
Volume4.Brick2.hostname: output omitted
Volume4.snapshot1.name: tv4-snap_GMT-2016.11.24-12.10.15
Volume4.snapshot1.id: 2eea76ae-c99f-4128-b5c0-3233048312f2
Volume4.snapshot1.time: 2016-11-24 12:10:15
Volume4.snapshot1.status: in_use
Volume4.snap_count: 1
Volume4.stripe_count: 1
Volume4.subvol_count: 2
Volume4.arbiter_count: 0
Volume4.disperse_count: 0
Volume4.redundancy_count: 0
Volume4.quorum_status: not_applicable
Volume4.snapd_svc.online_status: Offline
Volume4.snapd_svc.inited: True
Volume4.rebalance.id: 00000000-0000-0000-0000-000000000000
Volume4.rebalance.status: not_started
Volume4.rebalance.failures: 0
Volume4.rebalance.skipped: 0
Volume4.rebalance.lookedup: 0
Volume4.rebalance.files: 0
Volume4.rebalance.data: 0
Volume4.rebalance.data: 0
[Volume4.options]
features.uss: on
transport.address-family: inet
performance.readdir-ahead: on
nfs.disable: on


[Services]
svc1.name: glustershd
svc1.online_status: Offline


svc2.name: nfs
```

```
svc2.online_status: Offline


svc3.name: bitd
svc3.online_status: Offline


svc4.name: scrub
svc4.online_status: Offline


svc5.name: quotad
svc5.online_status: Offline


[Misc]
Base port: 49152
Last allocated port: 49154
```

Invocation of the **`gluster get-state volumeoptions`** lists all volume options irrespective of whether the volume option has been explicitly set or not.

Example Output for `gluster get-state volumeoptions`:

```
# gluster get-state volumeoptions
glusterd state dumped to /var/run/gluster/glusterd_state_timestamp
```

View the file using the **`cat`** ***`state_dump_file_path`*** command:

```
[Volume Options]
Volume1.name: Distvol
Volume1.options.count: 309
Volume1.options.value309: 0
Volume1.options.key309: cluster.max-bricks-per-process
Volume1.options.value308: off
Volume1.options.key308: cluster.brick-multiplex
Volume1.options.value307: on
Volume1.options.key307: disperse.optimistic-change-log
Volume1.options.value306: 60
Volume1.options.key306: performance.nl-cache-timeout
Volume1.options.value305: 10MB
Volume1.options.key305: performance.nl-cache-limit
Volume1.options.value304: false
Volume1.options.key304: performance.nl-cache-positive-entry
Volume1.options.value303: 10MB
Volume1.options.key303: performance.rda-cache-limit
Volume1.options.value302: 128KB
Volume1.options.key302: performance.rda-high-wmark
Volume1.options.value301: 4096
Volume1.options.key301: performance.rda-low-wmark
Volume1.options.value300: 131072
Volume1.options.key300: performance.rda-request-size
Volume1.options.value299: off
Volume1.options.key299: performance.parallel-readdir
Volume1.options.value298: off
Volume1.options.key298: cluster.use-compound-fops
```

```
Volume1.options.value297: 1024
Volume1.options.key297: disperse.shd-wait-qlength
Volume1.options.value296: 1
Volume1.options.key296: disperse.shd-max-threads
Volume1.options.value295: off
Volume1.options.key295: cluster.use-compound-fops
Volume1.options.value294: no
Volume1.options.key294: cluster.granular-entry-heal
Volume1.options.value293: full
Volume1.options.key293: cluster.locking-scheme
Volume1.options.value292: 1024
Volume1.options.key292: cluster.shd-wait-qlength
Volume1.options.value291: 1
Volume1.options.key291: cluster.shd-max-threads
Volume1.options.value290: round-robin
Volume1.options.key290: disperse.read-policy
Volume1.options.value289: on
Volume1.options.key289: dht.force-readdirp
Volume1.options.value288: 600
Volume1.options.key288: cluster.heal-timeout
Volume1.options.value287: 128
Volume1.options.key287: disperse.heal-wait-qlength
Volume1.options.value286: 8
Volume1.options.key286: disperse.background-heals
Volume1.options.value285: 60
Volume1.options.key285: features.lease-lock-recall-timeout
Volume1.options.value284: off
Volume1.options.key284: features.leases
Volume1.options.value283: 60
Volume1.options.key283: features.cache-invalidation-timeout
Volume1.options.value282: off
Volume1.options.key282: features.cache-invalidation
Volume1.options.value281: 120
Volume1.options.key281: features.expiry-time
Volume1.options.value280: false
Volume1.options.key280: features.scrub
Volume1.options.value279: biweekly
Volume1.options.key279: features.scrub-freq
Volume1.options.value278: lazy
Volume1.options.key278: features.scrub-throttle
Volume1.options.value277: 64MB
Volume1.options.key277: features.shard-block-size
Volume1.options.value276: off
Volume1.options.key276: features.shard
Volume1.options.value275: off
Volume1.options.key275: ganesha.enable
Volume1.options.value274: (null)
Volume1.options.key274: client.bind-insecure
Volume1.options.value273: no
Volume1.options.key273: cluster.quorum-reads
Volume1.options.value272: enable
Volume1.options.key272: cluster.disperse-self-heal-daemon
Volume1.options.value271: off
Volume1.options.key271: locks.mandatory-locking
Volume1.options.value270: off
Volume1.options.key270: locks.trace
```

```
Volume1.options.value269: 25000
Volume1.options.key269: features.ctr-sql-db-wal-autocheckpoint
Volume1.options.value268: 12500
Volume1.options.key268: features.ctr-sql-db-cachesize
Volume1.options.value267: 300
Volume1.options.key267: features.ctr_lookupheal_inode_timeout
Volume1.options.value266: 300
Volume1.options.key266: features.ctr_lookupheal_link_timeout
Volume1.options.value265: off
Volume1.options.key265: features.ctr_link_consistency
Volume1.options.value264: off
Volume1.options.key264: features.ctr-record-metadata-heat
Volume1.options.value263: off
Volume1.options.key263: features.record-counters
Volume1.options.value262: off
Volume1.options.key262: features.ctr-enabled
Volume1.options.value261: 100
Volume1.options.key261: cluster.tier-query-limit
Volume1.options.value260: 10000
Volume1.options.key260: cluster.tier-max-files
Volume1.options.value259: 4000
Volume1.options.key259: cluster.tier-max-mb
Volume1.options.value258: 0
Volume1.options.key258: cluster.tier-max-promote-file-size
Volume1.options.value257: cache
Volume1.options.key257: cluster.tier-mode
Volume1.options.value256: 75
Volume1.options.key256: cluster.watermark-low
Volume1.options.value255: 90
Volume1.options.key255: cluster.watermark-hi
Volume1.options.value254: 3600
Volume1.options.key254: cluster.tier-demote-frequency
Volume1.options.value253: 120
Volume1.options.key253: cluster.tier-promote-frequency
Volume1.options.value252: off
Volume1.options.key252: cluster.tier-pause
Volume1.options.value251: 0
Volume1.options.key251: cluster.read-freq-threshold
Volume1.options.value250: 0
Volume1.options.key250: cluster.write-freq-threshold
Volume1.options.value249: disable
Volume1.options.key249: cluster.enable-shared-storage
Volume1.options.value248: off
Volume1.options.key248: features.trash-internal-op
Volume1.options.value247: 5MB
Volume1.options.key247: features.trash-max-filesize
Volume1.options.value246: (null)
Volume1.options.key246: features.trash-eliminate-path
Volume1.options.value245: .trashcan
Volume1.options.key245: features.trash-dir
Volume1.options.value244: off
Volume1.options.key244: features.trash
Volume1.options.value243: 120
Volume1.options.key243: features.barrier-timeout
Volume1.options.value242: disable
Volume1.options.key242: features.barrier
```

```
Volume1.options.value241: off
Volume1.options.key241: changelog.capture-del-path
Volume1.options.value240: 120
Volume1.options.key240: changelog.changelog-barrier-timeout
Volume1.options.value239: 5
Volume1.options.key239: changelog.fsync-interval
Volume1.options.value238: 15
Volume1.options.key238: changelog.rollover-time
Volume1.options.value237: ascii
Volume1.options.key237: changelog.encoding
Volume1.options.value236: (null)
Volume1.options.key236: changelog.changelog-dir
Volume1.options.value235: off
Volume1.options.key235: changelog.changelog
Volume1.options.value234: 0
Volume1.options.key234: cluster.server-quorum-ratio
Volume1.options.value233: off
Volume1.options.key233: cluster.server-quorum-type
Volume1.options.value232: off
Volume1.options.key232: storage.bd-aio
Volume1.options.value231: off
Volume1.options.key231: storage.build-pgfid
Volume1.options.value230: 30
Volume1.options.key230: storage.health-check-interval
Volume1.options.value229: off
Volume1.options.key229: storage.node-uuid-pathinfo
Volume1.options.value228: -1
Volume1.options.key228: storage.owner-gid
Volume1.options.value227: -1
Volume1.options.key227: storage.owner-uid
Volume1.options.value226: 0
Volume1.options.key226: storage.batch-fsync-delay-usec
Volume1.options.value225: reverse-fsync
Volume1.options.key225: storage.batch-fsync-mode
Volume1.options.value224: off
Volume1.options.key224: storage.linux-aio
Volume1.options.value223: 180
Volume1.options.key223: features.auto-commit-period
Volume1.options.value222: relax
Volume1.options.key222: features.retention-mode
Volume1.options.value221: 120
Volume1.options.key221: features.default-retention-period
Volume1.options.value220: off
Volume1.options.key220: features.worm-file-level
Volume1.options.value219: off
Volume1.options.key219: features.worm
Volume1.options.value218: off
Volume1.options.key218: features.read-only
Volume1.options.value217: (null)
Volume1.options.key217: nfs.auth-cache-ttl-sec
Volume1.options.value216: (null)
Volume1.options.key216: nfs.auth-refresh-interval-sec
Volume1.options.value215: (null)
Volume1.options.key215: nfs.exports-auth-enable
Volume1.options.value214: on
Volume1.options.key214: nfs.rdirplus
```

```
Volume1.options.value213: (1 * 1048576ULL)
Volume1.options.key213: nfs.readdir-size
Volume1.options.value212: (1 * 1048576ULL)
Volume1.options.key212: nfs.write-size
Volume1.options.value211: (1 * 1048576ULL)
Volume1.options.key211: nfs.read-size
Volume1.options.value210: 0x20000
Volume1.options.key210: nfs.drc-size
Volume1.options.value209: off
Volume1.options.key209: nfs.drc
Volume1.options.value208: off
Volume1.options.key208: nfs.server-aux-gids
Volume1.options.value207: /sbin/rpc.statd
Volume1.options.key207: nfs.rpc-statd
Volume1.options.value206: /var/lib/glusterd/nfs/rmtab
Volume1.options.key206: nfs.mount-rmtab
Volume1.options.value205: off
Volume1.options.key205: nfs.mount-udp
Volume1.options.value204: ON
Volume1.options.key204: nfs.acl
Volume1.options.value203: on
Volume1.options.key203: nfs.nlm
Volume1.options.value202: ON
Volume1.options.key202: nfs.disable
Volume1.options.value201:
Volume1.options.key201: nfs.export-dir
Volume1.options.value200: read-write
Volume1.options.key200: nfs.volume-access
Volume1.options.value199: off
Volume1.options.key199: nfs.trusted-write
Volume1.options.value198: off
Volume1.options.key198: nfs.trusted-sync
Volume1.options.value197: off
Volume1.options.key197: nfs.ports-insecure
Volume1.options.value196: none
Volume1.options.key196: nfs.rpc-auth-reject
Volume1.options.value195: all
Volume1.options.key195: nfs.rpc-auth-allow
Volume1.options.value194: on
Volume1.options.key194: nfs.rpc-auth-null
Volume1.options.value193: on
Volume1.options.key193: nfs.rpc-auth-unix
Volume1.options.value192: 2049
Volume1.options.key192: nfs.port
Volume1.options.value191: 16
Volume1.options.key191: nfs.outstanding-rpc-limit
Volume1.options.value190: on
Volume1.options.key190: nfs.register-with-portmap
Volume1.options.value189: off
Volume1.options.key189: nfs.dynamic-volumes
Volume1.options.value188: off
Volume1.options.key188: nfs.addr-namelookup
Volume1.options.value187: on
Volume1.options.key187: nfs.export-volumes
Volume1.options.value186: on
Volume1.options.key186: nfs.export-dirs
```

```
Volume1.options.value185: 15
Volume1.options.key185: nfs.mem-factor
Volume1.options.value184: ON
Volume1.options.key184: nfs.enable-ino32
Volume1.options.value183: (null)
Volume1.options.key183: debug.error-fops
Volume1.options.value182: off
Volume1.options.key182: debug.random-failure
Volume1.options.value181: (null)
Volume1.options.key181: debug.error-number
Volume1.options.value180: (null)
Volume1.options.key180: debug.error-failure
Volume1.options.value179: off
Volume1.options.key179: debug.error-gen
Volume1.options.value178: (null)
Volume1.options.key178: debug.include-ops
Volume1.options.value177: (null)
Volume1.options.key177: debug.exclude-ops
Volume1.options.value176: no
Volume1.options.key176: debug.log-file
Volume1.options.value175: no
Volume1.options.key175: debug.log-history
Volume1.options.value174: off
Volume1.options.key174: debug.trace
Volume1.options.value173: disable
Volume1.options.key173: features.bitrot
Volume1.options.value172: off
Volume1.options.key172: features.inode-quota
Volume1.options.value171: off
Volume1.options.key171: features.quota
Volume1.options.value170: off
Volume1.options.key170: geo-replication.ignore-pid-check
Volume1.options.value169: off
Volume1.options.key169: geo-replication.ignore-pid-check
Volume1.options.value168: off
Volume1.options.key168: geo-replication.indexing
Volume1.options.value167: off
Volume1.options.key167: geo-replication.indexing
Volume1.options.value166: off
Volume1.options.key166: features.quota-deem-statfs
Volume1.options.value165: 86400
Volume1.options.key165: features.alert-time
Volume1.options.value164: 5
Volume1.options.key164: features.hard-timeout
Volume1.options.value163: 60
Volume1.options.key163: features.soft-timeout
Volume1.options.value162: 80%
Volume1.options.key162: features.default-soft-limit
Volume1.options.value161: 0
Volume1.options.key161: features.quota-timeout
Volume1.options.value160: (null)
Volume1.options.key160: features.limit-usage
Volume1.options.value159: false
Volume1.options.key159: network.compression.debug
Volume1.options.value158: -1
Volume1.options.key158: network.compression.compression-level
```

```
Volume1.options.value157: 0
Volume1.options.key157: network.compression.min-size
Volume1.options.value156: 8
Volume1.options.key156: network.compression.mem-level
Volume1.options.value155: -15
Volume1.options.key155: network.compression.window-size
Volume1.options.value154: off
Volume1.options.key154: network.compression
Volume1.options.value153: off
Volume1.options.key153: features.show-snapshot-directory
Volume1.options.value152: .snaps
Volume1.options.key152: features.snapshot-directory
Volume1.options.value151: off
Volume1.options.key151: features.uss
Volume1.options.value150: false
Volume1.options.key150: performance.cache-invalidation
Volume1.options.value149: true
Volume1.options.key149: performance.force-readdirp
Volume1.options.value148: off
Volume1.options.key148: performance.nfs.io-threads
Volume1.options.value147: off
Volume1.options.key147: performance.nfs.stat-prefetch
Volume1.options.value146: off
Volume1.options.key146: performance.nfs.quick-read
Volume1.options.value145: off
Volume1.options.key145: performance.nfs.io-cache
Volume1.options.value144: off
Volume1.options.key144: performance.nfs.read-ahead
Volume1.options.value143: on
Volume1.options.key143: performance.nfs.write-behind
Volume1.options.value142: on
Volume1.options.key142: performance.client-io-threads
Volume1.options.value141: on
Volume1.options.key141: performance.stat-prefetch
Volume1.options.value140: off
Volume1.options.key140: performance.nl-cache
Volume1.options.value139: on
Volume1.options.key139: performance.open-behind
Volume1.options.value138: on
Volume1.options.key138: performance.quick-read
Volume1.options.value137: on
Volume1.options.key137: performance.io-cache
Volume1.options.value136: on
Volume1.options.key136: performance.readdir-ahead
Volume1.options.value135: on
Volume1.options.key135: performance.read-ahead
Volume1.options.value134: on
Volume1.options.key134: performance.write-behind
Volume1.options.value133: 10
Volume1.options.key133: transport.listen-backlog
Volume1.options.value132: inet
Volume1.options.key132: transport.address-family
Volume1.options.value131: (null)
Volume1.options.key131: ssl.ec-curve
Volume1.options.value130: (null)
Volume1.options.key130: ssl.dh-param
```

```
Volume1.options.value129: (null)
Volume1.options.key129: ssl.cipher-list
Volume1.options.value128: (null)
Volume1.options.key128: ssl.certificate-depth
Volume1.options.value127: (null)
Volume1.options.key127: ssl.crl-path
Volume1.options.value126: (null)
Volume1.options.key126: ssl.ca-list
Volume1.options.value125: (null)
Volume1.options.key125: ssl.private-key
Volume1.options.value124: (null)
Volume1.options.key124: ssl.own-cert
Volume1.options.value123: 1
Volume1.options.key123: server.event-threads
Volume1.options.value122: (null)
Volume1.options.key122: server.own-thread
Volume1.options.value121: 300
Volume1.options.key121: server.gid-timeout
Volume1.options.value120: on
Volume1.options.key120: client.send-gids
Volume1.options.value119: on
Volume1.options.key119: server.dynamic-auth
Volume1.options.value118: off
Volume1.options.key118: server.manage-gids
Volume1.options.value117: *
Volume1.options.key117: auth.ssl-allow
Volume1.options.value116: (null)
Volume1.options.key116: server.ssl
Volume1.options.value115: 10
Volume1.options.key115: features.grace-timeout
Volume1.options.value114: off
Volume1.options.key114: features.lock-heal
Volume1.options.value113: 64
Volume1.options.key113: server.outstanding-rpc-limit
Volume1.options.value112: /var/run/gluster
Volume1.options.key112: server.statedump-path
Volume1.options.value111: 65534
Volume1.options.key111: server.anongid
Volume1.options.value110: 65534
Volume1.options.key110: server.anonuid
Volume1.options.value109: off
Volume1.options.key109: server.root-squash
Volume1.options.value108: (null)
Volume1.options.key108: server.allow-insecure
Volume1.options.value107: (null)
Volume1.options.key107: transport.keepalive
Volume1.options.value106: (null)
Volume1.options.key106: auth.reject
Volume1.options.value105: *
Volume1.options.key105: auth.allow
Volume1.options.value104: 16384
Volume1.options.key104: network.inode-lru-limit
Volume1.options.value103: (null)
Volume1.options.key103: network.tcp-window-size
Volume1.options.value102: 42
Volume1.options.key102: network.ping-timeout
```

```
Volume1.options.value101: 2
Volume1.options.key101: client.event-threads
Volume1.options.value100: disable
Volume1.options.key100: network.remote-dio
Volume1.options.value99: 10
Volume1.options.key99: features.grace-timeout
Volume1.options.value98: off
Volume1.options.key98: features.lock-heal
Volume1.options.value97: (null)
Volume1.options.key97: network.tcp-window-size
Volume1.options.value96: 42
Volume1.options.key96: network.ping-timeout
Volume1.options.value95: 1800
Volume1.options.key95: network.frame-timeout
Volume1.options.value94: 4096
Volume1.options.key94: encryption.block-size
Volume1.options.value93: 256
Volume1.options.key93: encryption.data-key-size
Volume1.options.value92: (null)
Volume1.options.key92: encryption.master-key
Volume1.options.value90: false
Volume1.options.key90: performance.cache-samba-metadata
Volume1.options.value89: true
Volume1.options.key89: performance.cache-swift-metadata
Volume1.options.value88: 1
Volume1.options.key88: performance.md-cache-timeout
Volume1.options.value87: 4
Volume1.options.key87: performance.read-ahead-page-count
Volume1.options.value86: no
Volume1.options.key86: performance.read-after-open
Volume1.options.value85: yes
Volume1.options.key85: performance.lazy-open
Volume1.options.value84: off
Volume1.options.key84: performance.nfs.strict-write-ordering
Volume1.options.value83: off
Volume1.options.key83: performance.strict-write-ordering
Volume1.options.value82: off
Volume1.options.key82: performance.nfs.strict-o-direct
Volume1.options.value81: off
Volume1.options.key81: performance.strict-o-direct
Volume1.options.value80: 1MB
Volume1.options.key80: performance.nfs.write-behind-window-size
Volume1.options.value79: off
Volume1.options.key79: performance.resync-failed-syncs-after-fsync
Volume1.options.value78: 1MB
Volume1.options.key78: performance.write-behind-window-size
Volume1.options.value77: on
Volume1.options.key77: performance.nfs.flush-behind
Volume1.options.value76: on
Volume1.options.key76: performance.flush-behind
Volume1.options.value75: 128MB
Volume1.options.key75: performance.cache-size
Volume1.options.value74: 0
Volume1.options.key74: performance.least-rate-limit
Volume1.options.value73: on
Volume1.options.key73: performance.enable-least-priority
```

```
Volume1.options.value72: 1
Volume1.options.key72: performance.least-prio-threads
Volume1.options.value71: 16
Volume1.options.key71: performance.low-prio-threads
Volume1.options.value70: 16
Volume1.options.key70: performance.normal-prio-threads
Volume1.options.value69: 16
Volume1.options.key69: performance.high-prio-threads
Volume1.options.value68: 16
Volume1.options.key68: performance.io-thread-count
Volume1.options.value67: 32MB
Volume1.options.key67: performance.cache-size
Volume1.options.value66:
Volume1.options.key66: performance.cache-priority
Volume1.options.value65: 1
Volume1.options.key65: performance.cache-refresh-timeout
Volume1.options.value64: 0
Volume1.options.key64: performance.cache-min-file-size
Volume1.options.value63: 0
Volume1.options.key63: performance.cache-max-file-size
Volume1.options.value62: 86400
Volume1.options.key62: diagnostics.stats-dnscache-ttl-sec
Volume1.options.value61: 65535
Volume1.options.key61: diagnostics.fop-sample-buf-size
Volume1.options.value60: 0
Volume1.options.key60: diagnostics.fop-sample-interval
Volume1.options.value59: 0
Volume1.options.key59: diagnostics.stats-dump-interval
Volume1.options.value58: 120
Volume1.options.key58: diagnostics.client-log-flush-timeout
Volume1.options.value57: 120
Volume1.options.key57: diagnostics.brick-log-flush-timeout
Volume1.options.value56: 5
Volume1.options.key56: diagnostics.client-log-buf-size
Volume1.options.value55: 5
Volume1.options.key55: diagnostics.brick-log-buf-size
Volume1.options.value54: (null)
Volume1.options.key54: diagnostics.client-log-format
Volume1.options.value53: (null)
Volume1.options.key53: diagnostics.brick-log-format
Volume1.options.value52: (null)
Volume1.options.key52: diagnostics.client-logger
Volume1.options.value51: (null)
Volume1.options.key51: diagnostics.brick-logger
Volume1.options.value50: CRITICAL
Volume1.options.key50: diagnostics.client-sys-log-level
Volume1.options.value49: CRITICAL
Volume1.options.key49: diagnostics.brick-sys-log-level
Volume1.options.value48: INFO
Volume1.options.key48: diagnostics.client-log-level
Volume1.options.value47: INFO
Volume1.options.key47: diagnostics.brick-log-level
Volume1.options.value46: off
Volume1.options.key46: diagnostics.count-fop-hits
Volume1.options.value45: off
Volume1.options.key45: diagnostics.dump-fd-stats
```

```
Volume1.options.value44: off
Volume1.options.key44: diagnostics.latency-measurement
Volume1.options.value43: true
Volume1.options.key43: cluster.stripe-coalesce
Volume1.options.value42: 128KB
Volume1.options.key42: cluster.stripe-block-size
Volume1.options.value41: none
Volume1.options.key41: cluster.favorite-child-policy
Volume1.options.value40: 128
Volume1.options.key40: cluster.heal-wait-queue-length
Volume1.options.value39: no
Volume1.options.key39: cluster.consistent-metadata
Volume1.options.value38: on
Volume1.options.key38: cluster.ensure-durability
Volume1.options.value37: 1
Volume1.options.key37: cluster.post-op-delay-secs
Volume1.options.value36: 1KB
Volume1.options.key36: cluster.self-heal-readdir-size
Volume1.options.value35: true
Volume1.options.key35: cluster.choose-local
Volume1.options.value34: (null)
Volume1.options.key34: cluster.quorum-count
Volume1.options.value33: none
Volume1.options.key33: cluster.quorum-type
Volume1.options.value32: on
Volume1.options.key32: disperse.eager-lock
Volume1.options.value31: on
Volume1.options.key31: cluster.eager-lock
Volume1.options.value30: (null)
Volume1.options.key30: cluster.data-self-heal-algorithm
Volume1.options.value29: on
Volume1.options.key29: cluster.metadata-change-log
Volume1.options.value28: on
Volume1.options.key28: cluster.data-change-log
Volume1.options.value27: 1
Volume1.options.key27: cluster.self-heal-window-size
Volume1.options.value26: 600
Volume1.options.key26: cluster.heal-timeout
Volume1.options.value25: on
Volume1.options.key25: cluster.self-heal-daemon
Volume1.options.value24: on
Volume1.options.key24: cluster.entry-self-heal
Volume1.options.value23: on
Volume1.options.key23: cluster.data-self-heal
Volume1.options.value22: on
Volume1.options.key22: cluster.metadata-self-heal
Volume1.options.value21: 8
Volume1.options.key21: cluster.background-self-heal-count
Volume1.options.value20: 1
Volume1.options.key20: cluster.read-hash-mode
Volume1.options.value19: -1
Volume1.options.key19: cluster.read-subvolume-index
Volume1.options.value18: (null)
Volume1.options.key18: cluster.read-subvolume
Volume1.options.value17: on
Volume1.options.key17: cluster.entry-change-log
```

```
Volume1.options.value16: (null)
Volume1.options.key16: cluster.switch-pattern
Volume1.options.value15: on
Volume1.options.key15: cluster.weighted-rebalance
Volume1.options.value14: (null)
Volume1.options.key14: cluster.local-volume-name
Volume1.options.value13: off
Volume1.options.key13: cluster.lock-migration
Volume1.options.value12: normal
Volume1.options.key12: cluster.rebal-throttle
Volume1.options.value11: off
Volume1.options.key11: cluster.randomize-hash-range-by-gfid
Volume1.options.value10: trusted.glusterfs.dht
Volume1.options.key10: cluster.dht-xattr-name
Volume1.options.value9: (null)
Volume1.options.key9: cluster.extra-hash-regex
Volume1.options.value8: (null)
Volume1.options.key8: cluster.rsync-hash-regex
Volume1.options.value7: off
Volume1.options.key7: cluster.readdir-optimize
Volume1.options.value6: (null)
Volume1.options.key6: cluster.subvols-per-directory
Volume1.options.value5: off
Volume1.options.key5: cluster.rebalance-stats
Volume1.options.value4: 5%
Volume1.options.key4: cluster.min-free-inodes
Volume1.options.value3: 10%
Volume1.options.key3: cluster.min-free-disk
Volume1.options.value2: off
Volume1.options.key2: cluster.lookup-optimize
Volume1.options.value1: on
Volume1.options.key1: cluster.lookup-unhashed
output truncated
```

19.7. Retrieving Current Volume Option Settings

Red Hat Gluster Storage allows storage administrators to retrieve the value of a specific volume option. You can also retrieve all the values of the volume options and all global options associated to a gluster volume. To retrieve the value of volume options, use the **`gluster volume get`** command. If a volume option is reconfigured for a volume, then the same value is displayed. If the volume option is not reconfigured, the default value is displayed.

The syntax is **`# gluster volume get <VOLNAME|all> <key|all>`**

19.7.1. Retrieving Value of a Specific Volume Option

To fetch the value of a specific volume option, execute the following command:

```
# gluster volume get <VOLNAME> <key>
```

Where,

VOLNAME: The volume name

key: The value of the volume option

For example:

```
# gluster volume get test-vol nfs.disable
 Option Value
 ------ -----
 nfs.disable on
```

19.7.2. Retrieving all Options of a Volume

To fetch the values of all the volume options, execute the following command:

```
# gluster volume get <VOLNAME> all
```

Where,

VOLNAME: The volume name

For example:

```
# gluster volume get test-vol all
 Option Value
 ------ -----
 cluster.lookup-unhashed on
 cluster.lookup-optimize off
 cluster.min-free-disk 10%
 cluster.min-free-inodes 5%
 cluster.rebalance-stats off
 cluster.subvols-per-directory (null)
 ....
```

19.7.3. Retrieving all Global Options

To fetch the values of all global options, execute the following command:

```
# gluster volume get all all
```

For example:

```
# gluster volume get all all

Option                                  Value
------                                  -----
cluster.server-quorum-ratio             51
cluster.enable-shared-storage           disable
cluster.op-version                      31305
cluster.max-op-version                  31305
cluster.brick-multiplex                 disable
cluster.max-bricks-per-process          250
cluster.daemon-log-level                INFO
```

19.8. Viewing complete volume state with statedump

The **statedump** subcommand writes out details of the current state of a specified process, including internal variables and other information that is useful for troubleshooting.

The command is used as follows:

```
# gluster volume statedump VOLNAME [[nfs|quotad]
[all|mem|iobuf|callpool|priv|fd|inode|history] | [client hostname:pid]]
```

19.8.1. Gathering information from the server

You can output all available state information, or limit statedump output to specific details, by using the statedump command with one of the following parameters.

all

Dumps all available state information.

mem

Dumps the memory usage and memory pool details of the bricks.

iobuf

Dumps iobuf details of the bricks.

priv

Dumps private information of loaded translators.

callpool

Dumps the pending calls of the volume.

fd

Dumps the open file descriptor tables of the volume.

inode

Dumps the inode tables of the volume.

history

Dumps the event history of the volume

For example, to write out all available information about the **data** volume, run the following command on the server:

```
# gluster volume statedump data all
```

If you only want to see details about the event history, run the following:

```
# gluster volume statedump data history
```

The ***nfs*** parameter is required to gather details about volumes shared via NFS. It can be combined with any of the above parameters to filter output.

```
# gluster volume statedump VOLNAME nfs all
```

The ***quotad*** parameter is required to gather details about the quota daemon. The following command writes out the state of the quota daemon across all nodes.

```
# gluster volume statedump VOLNAME quotad
```

If you need to see the state of a different process, such as the self-heal daemon, you can do so by running the following command using the process identifier of that process.

```
# kill -SIGUSR1 pid
```

19.8.2. Gathering information from the client

The **statedump** subcommand writes out details of the current state of a specified process, including internal variables and other information that is useful for troubleshooting.

To generate a statedump for client-side processes, using libgfapi, run the following command on a gluster node that is connected to the libgfapi application.

```
# gluster volume statedump VOLNAME client hostname:pid
```

Important

If you are using either NFS Ganesha or Samba service and you need to see the state of its clients, ensure that you use localhost instead of *hostname*. For example:

```
# gluster volume statedump VOLNAME client localhost:pid
```

If you need to get the state of glusterfs fuse mount process, you can do so by running the following command using the process identifier of that process.

```
# kill -SIGUSR1 pid
```

Important

If you have a gfapi based application and you need to see the state of its clients, ensure that the user running the gfapi application is a member of the **gluster** group. For example, if your gfapi application is run by user qemu, ensure that qemu is added to the gluster group by running the following command:

```
# usermod -a -G gluster qemu
```

19.8.3. Controlling statedump output location

Information is saved to the **/var/run/gluster** directory by default. Output files are named according to the following conventions:

- For brick processes, ***brick_path.brick_pid*.dump**
- For volume processes and **kill** command results, **glusterdump-*glusterd_pid*.dump.*timestamp***

To change where the output files of a particular volume are saved, use the ***server.statedump-path*** parameter, like so:

```
# gluster volume set VOLNAME server.statedump-path PATH
```

19.9. Displaying Volume Status

You can display the status information about a specific volume, brick, or all volumes, as needed. Status information can be used to understand the current status of the brick, NFS processes, self-heal daemon and overall file system. Status information can also be used to monitor and debug the volume information. You can view status of the volume along with the details:

- **detail** - Displays additional information about the bricks.
- **clients** - Displays the list of clients connected to the volume.
- **mem** - Displays the memory usage and memory pool details of the bricks.
- **inode** - Displays the inode tables of the volume.
- **fd** - Displays the open file descriptor tables of the volume.
- **callpool** - Displays the pending calls of the volume.

Setting Timeout Period

When you try to obtain information of a specific volume, the command may get timed out from the CLI if the originator **glusterd** takes longer than 120 seconds, the default time out, to aggregate the results from all the other **glusterd**s and report back to CLI.

You can use the **--timeout** option to ensure that the commands do not get timed out by 120 seconds.

For example,

```
# gluster volume status --timeout=500  VOLNAME  inode
```

It is recommended to use **--timeout** option when obtaining information about the inodes or clients or details as they frequently get timed out.

Display information about a specific volume using the following command:

# gluster volume status --timeout=*value_in_seconds* [all|*VOLNAME* [nfs | shd | *BRICKNAME*]] [detail |clients | mem | inode | fd |callpool]

For example, to display information about *test-volume*:

```
# gluster volume status test-volume
Status of volume: test-volume
Gluster process                        Port    Online   Pid
------------------------------------------------------------
```

```
Brick Server1:/rhgs/brick0/rep1        24010   Y       18474
Brick Server1:/rhgs/brick0/rep2        24011   Y       18479
NFS Server on localhost                38467   Y       18486
Self-heal Daemon on localhost          N/A     Y       18491
```

The self-heal daemon status will be displayed only for replicated volumes.

Display information about all volumes using the command:

```
# gluster volume status all
```

```
# gluster volume status all
Status of volume: test
Gluster process                       Port    Online   Pid
-----------------------------------------------------------
Brick Server1:/rhgs/brick0/test       24009   Y       29197
NFS Server on localhost               38467   Y       18486

Status of volume: test-volume
Gluster process                       Port    Online   Pid
------------------------------------------------------------
Brick Server1:/rhgs/brick0/rep1       24010   Y       18474
Brick Server1:/rhgs/brick0/rep2       24011   Y       18479
NFS Server on localhost               38467   Y       18486
Self-heal Daemon on localhost         N/A     Y       18491
```

Display additional information about the bricks using the command:

```
# gluster volume status VOLNAME detail
```

For example, to display additional information about the bricks of *test-volume*:

```
# gluster volume status test-volume detail
Status of volume: test-vol
------------------------------------------------------------------------
------
Brick                : Brick Server1:/rhgs/test
Port                 : 24012
Online               : Y
Pid                  : 18649
File System          : xfs
Device               : /dev/sda1
Mount Options        :
rw,relatime,user_xattr,acl,commit=600,barrier=1,data=ordered
Inode Size           : 256
Disk Space Free      : 22.1GB
Total Disk Space     : 46.5GB
Inode Count          : 3055616
Free Inodes          : 2577164
```

Detailed information is not available for NFS and the self-heal daemon.

Display the list of clients accessing the volumes using the command:

```
# gluster volume status VOLNAME clients
```

For example, to display the list of clients connected to *test-volume*:

```
# gluster volume status test-volume clients
Brick : Server1:/rhgs/brick0/1
Clients connected : 2
Hostname          Bytes Read   BytesWritten
--------          ---------    ------------
127.0.0.1:1013    776          676
127.0.0.1:1012    50440        51200
```

Client information is not available for the self-heal daemon.

Display the memory usage and memory pool details of the bricks on a volume using the command:

# **gluster volume status *VOLNAME* mem**

For example, to display the memory usage and memory pool details for the bricks on *test-volume*:

```
# gluster volume status test-volume mem
Memory status for volume : test-volume
----------------------------------------------
Brick : Server1:/rhgs/brick0/1
Mallinfo
--------
Arena    : 434176
Ordblks  : 2
Smblks   : 0
Hblks    : 12
Hblkhd   : 40861696
Usmblks  : 0
Fsmblks  : 0
Uordblks : 332416
Fordblks : 101760
Keepcost : 100400

Mempool Stats
-------------
Name                               HotCount ColdCount PaddedSizeof
AllocCount MaxAlloc
----                               -------- --------- ------------ ------
---- --------
test-volume-server:fd_t                0     16384           92         57
5
test-volume-server:dentry_t           59       965           84         59
59
test-volume-server:inode_t            60       964          148         60
60
test-volume-server:rpcsvc_request_t    0       525         6372        351
2
glusterfs:struct saved_frame           0      4096          124          2
2
glusterfs:struct rpc_req               0      4096         2236          2
2
glusterfs:rpcsvc_request_t             1       524         6372          2
1
glusterfs:call_stub_t                  0      1024         1220        288
1
```

```
glusterfs:call_stack_t                 0      8192         2084        290
2
glusterfs:call_frame_t                 0     16384          172       1728
6
```

Display the inode tables of the volume using the command:

```
# gluster volume status VOLNAME inode
```

For example, to display the inode tables of *test-volume*:

```
# gluster volume status test-volume inode
inode tables for volume test-volume
----------------------------------------------
Brick : Server1:/rhgs/brick0/1
Active inodes:
GFID                                            Lookups            Ref
IA type
----                                            -------            ---
-------
6f3fe173-e07a-4209-abb6-484091d75499                  1              9
2
370d35d7-657e-44dc-bac4-d6dd800ec3d3                  1              1
2

LRU inodes:
GFID                                            Lookups            Ref
IA type
----                                            -------            ---
-------
80f98abe-cdcf-4c1d-b917-ae564cf55763                  1              0
1
3a58973d-d549-4ea6-9977-9aa218f233de                  1              0
1
2ce0197d-87a9-451b-9094-9baa38121155                  1              0
2
```

Display the open file descriptor tables of the volume using the command:

```
# gluster volume status VOLNAME fd
```

For example, to display the open file descriptor tables of *test-volume*:

```
# gluster volume status test-volume fd

FD tables for volume test-volume
----------------------------------------------
Brick : Server1:/rhgs/brick0/1
Connection 1:
RefCount = 0  MaxFDs = 128  FirstFree = 4
FD Entry            PID                 RefCount            Flags
--------            ---                 --------            -----
0                   26311               1                   2
1                   26310               3                   2
2                   26310               1                   2
3                   26311               3                   2
```

```
Connection 2:
RefCount = 0  MaxFDs = 128  FirstFree = 0
No open fds

Connection 3:
RefCount = 0  MaxFDs = 128  FirstFree = 0
No open fds
```

FD information is not available for NFS and the self-heal daemon.

Display the pending calls of the volume using the command:

# gluster volume status *VOLNAME* callpool

Note, each call has a call stack containing call frames.

For example, to display the pending calls of *test-volume*:

```
# gluster volume status test-volume callpool

Pending calls for volume test-volume
----------------------------------------------
Brick : Server1:/rhgs/brick0/1
Pending calls: 2
Call Stack1
 UID    : 0
 GID    : 0
 PID    : 26338
 Unique : 192138
 Frames : 7
 Frame 1
  Ref Count   = 1
  Translator  = test-volume-server
  Completed   = No
 Frame 2
  Ref Count   = 0
  Translator  = test-volume-posix
  Completed   = No
  Parent      = test-volume-access-control
  Wind From   = default_fsync
  Wind To     = FIRST_CHILD(this)->fops->fsync
 Frame 3
  Ref Count   = 1
  Translator  = test-volume-access-control
  Completed   = No
  Parent      = repl-locks
  Wind From   = default_fsync
  Wind To     = FIRST_CHILD(this)->fops->fsync
 Frame 4
  Ref Count   = 1
  Translator  = test-volume-locks
  Completed   = No
  Parent      = test-volume-io-threads
  Wind From   = iot_fsync_wrapper
  Wind To     = FIRST_CHILD (this)->fops->fsync
 Frame 5
```

```
    Ref Count   = 1
    Translator  = test-volume-io-threads
    Completed   = No
    Parent      = test-volume-marker
    Wind From   = default_fsync
    Wind To     = FIRST_CHILD(this)->fops->fsync
   Frame 6
    Ref Count   = 1
    Translator  = test-volume-marker
    Completed   = No
    Parent      = /export/1
    Wind From   = io_stats_fsync
    Wind To     = FIRST_CHILD(this)->fops->fsync
   Frame 7
    Ref Count   = 1
    Translator  = /export/1
    Completed   = No
    Parent      = test-volume-server
    Wind From   = server_fsync_resume
    Wind To     = bound_xl->fops->fsync
```

19.10. Troubleshooting issues in the Red Hat Gluster Storage Trusted Storage Pool

19.10.1. Troubleshooting a network issue in the Red Hat Gluster Storage Trusted Storage Pool

When enabling the network components to communicate with Jumbo frames in a Red Hat Gluster Storage Trusted Storage Pool, ensure that all the network components such as switches, Red Hat Gluster Storage nodes etc are configured properly. Verify the network configuration by running the **ping** from one Red Hat Gluster Storage node to another.

If the nodes in the Red Hat Gluster Storage Trusted Storage Pool or any other network components are not configured to fully support Jumbo frames, the **ping** command times out and displays the following error:

```
# ping -s 1600 '-Mdo'
local error: Message too long, mtu=1500
```

Chapter 20. Managing Resource Usage

When Red Hat Gluster Storage is deployed on the same machine as other resource intensive software and services, it can be useful to limit the resources that glusterd attempts to use in order to avoid resource contention between processes.

Procedure 20.1. Limiting glusterd resources on RHEL7 based Red Hat Gluster Storage

On Red Hat Gluster Storage 3.2 and higher deployments based on Red Hat Enterprise Linux 7, this can be configured using gdeploy. For more information, see Section 5.1.11, “Limiting Gluster Resources”.

On earlier versions of Red Hat Gluster Storage, it is necessary to manually configure a control group slice for the glusterd service in order to manage glusterd's access to system resources.

1. Stop all gluster processes

   ```
   # systemctl stop glusterd
   ```

   Important

   If **glusterd** crashes, there is no functionality impact to this crash as it occurs during the shutdown. For more information, see Section 24.3, “Resolving **glusterd** Crash”

2. Create a service configuration directory for glusterd

   ```
   # mkdir /etc/systemd/system/glusterd.service.d
   ```

3. Create a service configuration file

   ```
   # echo "[Service]
   CPUAccounting=yes
   Slice=glusterfs.slice" >> /etc/systemd/system/glusterd.service.d/99-
   cpu.conf
   ```

4. Create a slice file

   The following defines a slice that sets ***CPUQuota*** to the recommended value of **400%** (four cores).

   ```
   # echo "[Slice]
   CPUQuota=400%" >> /etc/systemd/system/glusterfs.slice
   ```

   You can alter the percentage to suit your environment by editing the value in the slice file:

   ```
   # systemctl set-property glusterfs.slice CPUQuota=value
   ```

5. Restart the system daemon

   ```
   # systemctl daemon-reload
   ```

6. Start gluster processes

```
# systemctl start glusterd
```

For more information about configuring resource management on Red Hat Enterprise Linux 7, see the *Resource Management Guide*: https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html-single/Resource_Management_Guide/index.html#sec-What_are_Control_Groups

Resource management works differently on Red Hat Enterprise Linux 6. See the Red Hat Enterprise Linux 6 *Resource Management Guide* for details: https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Resource_Management_Guide/ch-Using_Control_Groups.html

Procedure 20.2. Controlling CPU Usage for a Gluster Daemon

The **`control-cpu-load`** script provides a utility to control CPU utilization for any Gluster daemon by using the cgroup framework to configure CPU quota for a process.

1. Navigate to the scripts folder by using the following command:

   ```
   # cd /usr/share/glusterfs/scripts
   ```

2. Determine the PID of the required gluster daemon by using the following command:

   ```
   # ps -aef | grep daemon_name
   ```

   The output will be in the following format:

   ```
   root      1565...output omitted...grep --color=auto daemon_name
   ```

   In this output, **1565** represents the PID of the daemon service. PIDs are unlikely to be the same on different systems, or for different instances of the daemon, so ensure that you check for the relevant PID every time you perform this process.

3. Execute the **`control-cpu-load`** script by using the following command:

   ```
   # sh control-cpu-load.sh
   ```

4. When the system prompts you with the following input, type the PID of the daemon acquired from the previous step and press **`Enter`**:

   ```
   [root@XX-XX scripts]# sh control-cpu-load.sh
   Enter gluster daemon pid for which you want to control CPU.
   1565
   ```

5. When the system prompts you with the following input, type **`y`** and press **`Enter`**:

   ```
   If you want to continue the script to attach 1565 with new
   cgroup_gluster_1565 cgroup Press (y/n)?
   ```

6. When the system prompts the following notification, enter the required quota value to be assigned to the daemon and press **`Enter`**:

   ```
   Creating child cgroup directory 'cgroup_gluster_1565 cgroup' for
   daemon_name.service.
   Enter quota value in range [10,100]:
   ```

```
25
```

In this example, quota value for the daemon service is set to **25**.

The system displays the following message once the quota value has been successfully set:

```
Entered quota value is 25
Setting 25000 to cpu.cfs_quota_us for gluster_cgroup.
Tasks are attached successfully specific to 1565 to
cgroup_gluster_1565.
```

Important

Perform this procedure every time the daemon is restarted and has a new daemon PID.

Procedure 20.3. Controlling memory usage for a Gluster daemon

The **control-mem** script provides a utility to control memory utilization for any Gluster daemon by using the cgroup framework to configure memory limit for a process.

1. Navigate to the scripts folder by using the following command:

   ```
   # cd /usr/share/glusterfs/scripts
   ```

2. Determine the PID of the required gluster daemon by using the following command:

   ```
   # ps -aef | grep daemon_name
   ```

   The output will be in the following format:

   ```
   root      1565     1  0 Feb05 ?        00:09:17 /usr/sbin/glusterfs -
   s localhost --volfile-id gluster/daemon_name -p
   /var/run/gluster/daemon_name/daemon_name.pid -l
   /var/log/glusterfs/daemon_name.log -S
   /var/run/gluster/ed49b959a0dc9b2185913084e3b2b339.socket --xlator-
   option *replicate*.node-uuid=13dbfa1e-ebbf-4cee-a1ac-ca6763903c55
     root     16766 14420  0 19:00 pts/0    00:00:00 grep --color=auto
   daemon_name
   ```

   In this output, **1565** represents the PID of the daemon service.

3. Execute the **control-mem** script by using the following command:

   ```
   # sh control-mem.sh
   ```

4. When the system prompts for the following input, type the PID of the daemon acquired from the previous step and press **Enter**:

   ```
   [root@XX-XX scripts]# sh control-mem.sh
   Enter gluster daemon pid for which you want to control CPU.
   1565
   ```

In this example, **1565** represents the PID of the daemon service. The PID of the daemon services can vary from system to system.

5. When the system prompts for the following input, type **y** and press **Enter**:

```
If you want to continue the script to attach daeomon with new cgroup.
Press (y/n)?
```

The system prompts the following notification:

```
Creating child cgroup directory 'cgroup_gluster_1565 cgroup' for
daemon_name.service.
```

6. When the system prompts for the following input, enter the required memory value to be assigned to the daemon and press **Enter**:

```
Enter Memory value in Mega bytes [100,8000000000000]:
```

In this example, the memory value is set to **5000**. The system prompts the following message once the memory value has been successfully set:

```
Entered memory limit value is 5000.
Setting 5242880000 to memory.limit_in_bytes for
/sys/fs/cgroup/memory/system.slice/daemon_name.service/cgroup_gluster_
1565.
Tasks are attached successfully specific to 1565 to
cgroup_gluster_1565.
```

Important

Perform this procedure every time the daemon is restarted and has a new daemon PID.

Chapter 21. Tuning for Performance

This chapter provides information on configuring Red Hat Gluster Storage and explains clear and simple activities that can improve system performance.

21.1. Disk Configuration

Red Hat Gluster Storage supports JBOD (Just a Bunch of Disks) and hardware RAID storage.

21.1.1. Hardware RAID

The RAID levels that are most commonly recommended are RAID 6 and RAID 10. RAID 6 provides better space efficiency, good read performance and good performance for sequential writes to large files.

When configured across 12 disks, RAID 6 can provide ~40% more storage space in comparison to RAID 10, which has a 50% reduction in capacity. However, RAID 6 performance for small file writes and random writes tends to be lower than RAID 10. If the workload is strictly small files, then RAID 10 is the optimal configuration.

An important parameter in hardware RAID configuration is the stripe unit size. With thin provisioned disks, the choice of RAID stripe unit size is closely related to the choice of thin-provisioning chunk size.

For RAID 10, a stripe unit size of 256 KiB is recommended.

For RAID 6, the stripe unit size must be chosen such that the full stripe size (stripe unit * number of data disks) is between 1 MiB and 2 MiB, preferably in the lower end of the range. Hardware RAID controllers usually allow stripe unit sizes that are a power of 2. For RAID 6 with 12 disks (10 data disks), the recommended stripe unit size is 128KiB.

21.1.2. JBOD

In the JBOD configuration, physical disks are not aggregated into RAID devices, but are visible as separate disks to the operating system. This simplifies system configuration by not requiring a hardware RAID controller.

If disks on the system are connected through a hardware RAID controller, refer to the RAID controller documentation on how to create a JBOD configuration; typically, JBOD is realized by exposing **`raw`** drives to the operating system using a **`pass-through`** mode.

In the JBOD configuration, a single physical disk serves as storage for a Red Hat Gluster Storage brick.

JBOD configurations support up to 36 disks per node with dispersed volumes and three-way replication.

21.2. Brick Configuration

Format bricks using the following configurations to enhance performance:

Procedure 21.1. Brick Configuration

1. **LVM layer**

   The steps for creating a brick from a physical device is listed below. An outline of steps for creating multiple bricks on a physical device is listed as *Example - Creating multiple bricks on a physical device* below.

- **Creating the Physical Volume**

  The **pvcreate** command is used to create the physical volume. The Logical Volume Manager can use a portion of the physical volume for storing its metadata while the rest is used as the data portion.Align the I/O at the Logical Volume Manager (LVM) layer using **--dataalignment** option while creating the physical volume.

  The command is used in the following format:

  ```
  # pvcreate --dataalignment alignment_value disk
  ```

  For JBOD, use an alignment value of **256K**.

  In case of hardware RAID, the *alignment_value* should be obtained by multiplying the RAID stripe unit size with the number of data disks. If 12 disks are used in a RAID 6 configuration, the number of data disks is 10; on the other hand, if 12 disks are used in a RAID 10 configuration, the number of data disks is 6.

  For example, the following command is appropriate for 12 disks in a RAID 6 configuration with a stripe unit size of 128 KiB:

  ```
  # pvcreate --dataalignment 1280k disk
  ```

  The following command is appropriate for 12 disks in a RAID 10 configuration with a stripe unit size of 256 KiB:

  ```
  # pvcreate --dataalignment 1536k disk
  ```

  To view the previously configured physical volume settings for **--dataalignment**, run the following command:

  ```
  # pvs -o +pe_start disk
    PV         VG   Fmt  Attr PSize PFree 1st PE
    /dev/sdb        lvm2 a--  9.09t 9.09t   1.25m
  ```

- **Creating the Volume Group**

  The volume group is created using the **vgcreate** command.

  For hardware RAID, in order to ensure that logical volumes created in the volume group are aligned with the underlying RAID geometry, it is important to use the **-- physicalextentsize** option. Execute the **vgcreate** command in the following format:

  ```
  # vgcreate --physicalextentsize extent_size VOLGROUP
  physical_volume
  ```

  The *extent_size* should be obtained by multiplying the RAID stripe unit size with the number of data disks. If 12 disks are used in a RAID 6 configuration, the number of data disks is 10; on the other hand, if 12 disks are used in a RAID 10 configuration, the number of data disks is 6.

  For example, run the following command for RAID-6 storage with a stripe unit size of 128 KB, and 12 disks (10 data disks):

  ```
  # vgcreate --physicalextentsize 1280k VOLGROUP physical_volume
  ```

In the case of JBOD, use the **vgcreate** command in the following format:

```
# vgcreate VOLGROUP physical_volume
```

- **Creating the Thin Pool**

  A thin pool provides a common pool of storage for thin logical volumes (LVs) and their snapshot volumes, if any.

  Execute the following commands to create a thin pool of a specific size:

  ```
  # lvcreate --thin VOLGROUP/POOLNAME --size POOLSIZE --chunksize
  CHUNKSIZE --poolmetadatasize METASIZE --zero n
  ```

  You can also create a thin pool of the maximum possible size for your device by executing the following command:

  ```
  # lvcreate --thin VOLGROUP/POOLNAME --extents 100%FREE --chunksize
  CHUNKSIZE --poolmetadatasize METASIZE --zero n
  ```

Recommended parameter values for thin pool creation

poolmetadatasize

Internally, a thin pool contains a separate metadata device that is used to track the (dynamically) allocated regions of the thin LVs and snapshots. The **poolmetadatasize** option in the above command refers to the size of the pool metadata device.

The maximum possible size for a metadata LV is 16 GiB. Red Hat Gluster Storage recommends creating the metadata device of the maximum supported size. You can allocate less than the maximum if space is a concern, but in this case you should allocate a minimum of 0.5% of the pool size.

> **Warning**
>
> If your metadata pool runs out of space, you cannot create data. This includes the data required to increase the size of the metadata pool or to migrate data away from a volume that has run out of metadata space. Monitor your metadata pool using the **lvs -o+metadata_percent** command and ensure that it does not run out of space.

chunksize

An important parameter to be specified while creating a thin pool is the chunk size,which is the unit of allocation. For good performance, the chunk size for the thin pool and the parameters of the underlying hardware RAID storage should be chosen so that they work well together.

For **JBOD**, use a thin pool chunk size of 256 KiB.

For **RAID 6** storage, the striping parameters should be chosen so that the full stripe size (stripe_unit size * number of data disks) is between 1 MiB and 2 MiB, preferably in the low end of the range. The thin pool chunk size should be chosen to match the RAID

6 full stripe size. Matching the chunk size to the full stripe size aligns thin pool allocations with RAID 6 stripes, which can lead to better performance. Limiting the chunk size to below 2 MiB helps reduce performance problems due to excessive copy-on-write when snapshots are used.

For example, for RAID 6 with 12 disks (10 data disks), stripe unit size should be chosen as 128 KiB. This leads to a full stripe size of 1280 KiB (1.25 MiB). The thin pool should then be created with the chunk size of 1280 KiB.

For **RAID 10** storage, the preferred stripe unit size is 256 KiB. This can also serve as the thin pool chunk size. Note that RAID 10 is recommended when the workload has a large proportion of small file writes or random writes. In this case, a small thin pool chunk size is more appropriate, as it reduces copy-on-write overhead with snapshots.

If the addressable storage on the device is smaller than the device itself, you need to adjust the recommended chunk size. Calculate the adjustment factor using the following formula:

```
adjustment_factor = device_size_in_tb /
(preferred_chunk_size_in_kb * 4 / 64 )
```

Round the adjustment factor up. Then calculate the new chunk size using the following:

```
chunk_size = preferred_chunk_size *
rounded_adjustment_factor
```

block zeroing

By default, the newly provisioned chunks in a thin pool are zeroed to prevent data leaking between different block devices. In the case of Red Hat Gluster Storage, where data is accessed via a file system, this option can be turned off for better performance with the **`--zero n`** option. Note that **`n`** does not need to be replaced.

The following example shows how to create the thin pool:

```
# lvcreate --thin VOLGROUP/thin_pool --size 2T --chunksize
1280k --poolmetadatasize 16G --zero n
```

You can also use **`--extents 100%FREE`** to ensure the thin pool takes up all available space once the metadata pool is created.

```
# lvcreate --thin VOLGROUP/thin_pool --extents 100%FREE --
chunksize 1280k --poolmetadatasize 16G --zero n
```

The following example shows how to create a 2 TB thin pool:

```
# lvcreate --thin VOLGROUP/thin_pool --size 2T --chunksize 1280k --
poolmetadatasize 16G --zero n
```

The following example creates a thin pool that takes up all remaining space once the metadata pool has been created.

```
# lvcreate --thin VOLGROUP/thin_pool --extents 100%FREE --chunksize
1280k --poolmetadatasize 16G --zero n
```

- **Creating a Thin Logical Volume**

  After the thin pool has been created as mentioned above, a thinly provisioned logical volume can be created in the thin pool to serve as storage for a brick of a Red Hat Gluster Storage volume.

  ```
  # lvcreate --thin --name LV_name --virtualsize LV_size
  VOLGROUP/thin_pool
  ```

- **Example - Creating multiple bricks on a physical device**

  The steps above (LVM Layer) cover the case where a single brick is being created on a physical device. This example shows how to adapt these steps when multiple bricks need to be created on a physical device.

  > **Note**
  >
  > In this following steps, we are assuming the following:
  > - Two bricks must be created on the same physical device
  > - One brick must be of size 4 TiB and the other is 2 TiB
  > - The device is **`/dev/sdb`**, and is a RAID-6 device with 12 disks
  > - The 12-disk RAID-6 device has been created according to the recommendations in this chapter, that is, with a stripe unit size of 128 KiB

  - Create a single physical volume using pvcreate

    ```
    # pvcreate --dataalignment 1280k /dev/sdb
    ```

  - Create a single volume group on the device

    ```
    # vgcreate --physicalextentsize 1280k vg1 /dev/sdb
    ```

  - Create a separate thin pool for each brick using the following commands:

    ```
    # lvcreate --thin vg1/thin_pool_1 --size 4T --chunksize 1280K
    --poolmetadatasize 16G --zero n
    ```

    ```
    # lvcreate --thin vg1/thin_pool_2 --size 2T --chunksize 1280K
    --poolmetadatasize 16G --zero n
    ```

    In the examples above, the size of each thin pool is chosen to be the same as the size of the brick that will be created in it. With thin provisioning, there are many possible ways of managing space, and these options are not discussed in this chapter.

  - Create a thin logical volume for each brick

    ```
    # lvcreate --thin --name lv1 --virtualsize 4T vg1/thin_pool_1
    ```

    ```
    # lvcreate --thin --name lv2 --virtualsize 2T vg1/thin_pool_2
    ```

  - Follow the *XFS Recommendations* (next step) in this chapter for creating and mounting filesystems for each of the thin logical volumes

```
# mkfs.xfs options /dev/vg1/lv1
```

```
# mkfs.xfs options /dev/vg1/lv2
```

```
# mount options /dev/vg1/lv1 mount_point_1
```

```
# mount options /dev/vg1/lv2 mount_point_2
```

2. **XFS Recommendataions**

- **XFS Inode Size**

  As Red Hat Gluster Storage makes extensive use of extended attributes, an XFS inode size of 512 bytes works better with Red Hat Gluster Storage than the default XFS inode size of 256 bytes. So, inode size for XFS must be set to 512 bytes while formatting the Red Hat Gluster Storage bricks. To set the inode size, you have to use -i size option with the **`mkfs.xfs`** command as shown in the following *Logical Block Size for the Directory* section.

- **XFS RAID Alignment**

  When creating an XFS file system, you can explicitly specify the striping parameters of the underlying storage in the following format:

  ```
  # mkfs.xfs other_options -d
  su=stripe_unit_size,sw=stripe_width_in_number_of_disks device
  ```

  For RAID 6, ensure that I/O is aligned at the file system layer by providing the striping parameters. For RAID 6 storage with 12 disks, if the recommendations above have been followed, the values must be as following:

  ```
  # mkfs.xfs other_options -d su=128k,sw=10 device
  ```

  For RAID 10 and JBOD, the **`-d su=<>,sw=<>`** option can be omitted. By default, XFS will use the thin-p chunk size and other parameters to make layout decisions.

- **Logical Block Size for the Directory**

  An XFS file system allows to select a logical block size for the file system directory that is greater than the logical block size of the file system. Increasing the logical block size for the directories from the default 4 K, decreases the directory I/O, which in turn improves the performance of directory operations. To set the block size, you need to use **`-n size`** option with the **`mkfs.xfs`** command as shown in the following example output.

  Following is the example output of RAID 6 configuration along with inode and block size options:

  ```
  # mkfs.xfs -f -i size=512 -n size=8192 -d su=128k,sw=10 logical
  volume
  meta-data=/dev/mapper/gluster-brick1 isize=512    agcount=32,
  agsize=37748736 blks
           =    sectsz=512   attr=2, projid32bit=0
  data     =     bsize=4096   blocks=1207959552, imaxpct=5
           =    sunit=32     swidth=320 blks
  ```

```
naming   = version 2   bsize=8192   ascii-ci=0
log      =internal log   bsize=4096   blocks=521728, version=2
         =     sectsz=512   sunit=32 blks, lazy-count=1
realtime =none    extsz=4096   blocks=0, rtextents=0
```

- **Allocation Strategy**

  inode32 and inode64 are two most common allocation strategies for XFS. With inode32 allocation strategy, XFS places all the inodes in the first 1 TiB of disk. With larger disk, all the inodes would be stuck in first 1 TiB. inode32 allocation strategy is used by default.

  With inode64 mount option inodes would be replaced near to the data which would be minimize the disk seeks.

  To set the allocation strategy to inode64 when file system is being mounted, you need to use **`-o inode64`** option with the **`mount`** command as shown in the following **Access Time** section.

- **Access Time**

  If the application does not require to update the access time on files, than file system must always be mounted with **`noatime`** mount option. For example:

  ```
  # mount -t xfs -o inode64,noatime <logical volume> <mount point>
  ```

  This optimization improves performance of small-file reads by avoiding updates to the XFS inodes when files are read.

  ```
  /etc/fstab entry for option E + F
   <logical volume> <mount point>xfs     inode64,noatime   0 0
  ```

- **Allocation groups**

  Each XFS file system is partitioned into regions called allocation groups. Allocation groups are similar to the block groups in ext3, but allocation groups are much larger than block groups and are used for scalability and parallelism rather than disk locality. The default allocation for an allocation group is 1 TiB.

  Allocation group count must be large enough to sustain the concurrent allocation workload. In most of the cases allocation group count chosen by **`mkfs.xfs`** command would give the optimal performance. Do not change the allocation group count chosen by **`mkfs.xfs`**, while formatting the file system.

- **Percentage of space allocation to inodes**

  If the workload is very small files (average file size is less than 10 KB), then it is recommended to set **`maxpct`** value to **`10`**, while formatting the file system.

3. **Performance tuning option in Red Hat Gluster Storage**

   A tuned profile is designed to improve performance for a specific use case by tuning system parameters appropriately. Red Hat Gluster Storage includes tuned profiles tailored for its workloads. These profiles are available in both Red Hat Enterprise Linux 6 and Red Hat Enterprise Linux 7.

   Table 21.1. Recommended Profiles for Different Workloads

Workload	Profile Name
Large-file, sequential I/O workloads	**`rhgs-sequential-io`**

Workload	Profile Name
Small-file workloads	**`rhgs-random-io`**
Random I/O workloads	**`rhgs-random-io`**

Earlier versions of Red Hat Gluster Storage on Red Hat Enterprise Linux 6 recommended tuned profiles **`rhs-high-throughput`** and **`rhs-virtualization`**. These profiles are still available on Red Hat Enterprise Linux 6. However, switching to the new profiles is recommended.

To apply tunings contained in the tuned profile, run the following command after creating a Red Hat Gluster Storage volume.

```
# tuned-adm profile profile-name
```

For example:

```
# tuned-adm profile rhgs-sequential-io
```

4. **Writeback Caching**

   For small-file and random write performance, we strongly recommend writeback cache, that is, non-volatile random-access memory (NVRAM) in your storage controller. For example, normal Dell and HP storage controllers have it. Ensure that NVRAM is enabled, that is, the battery is working. Refer your hardware documentation for details on enabling NVRAM.

   Do not enable writeback caching in the disk drives, this is a policy where the disk drive considers the write is complete before the write actually made it to the magnetic media (platter). As a result, the disk write cache might lose its data during a power failure or even loss of metadata leading to file system corruption.

21.2.1. Many Bricks per Node

By default, for every brick configured on a Red Hat Gluster Storage server node, one process is created and one port is consumed. If you have a large number of bricks configured on a single server, enabling brick multiplexing reduces port and memory consumption by allowing compatible bricks to use the same process and port. Red Hat recommends restarting all volumes after enabling or disabling brick multiplexing.

As of Red Hat Gluster Storage 3.4, brick multiplexing is supported only for Container-Native Storage (CNS) and Container-Ready Storage(CRS) use cases.

Configuring Brick Multiplexing

1. Set ***`cluster.brick-multiplex`*** to **on**. This option affects all volumes.

   ```
   # gluster volume set all cluster.brick-multiplex on
   ```

2. Increase the limits for socket traffic.

   ```
   # gluster volume set all transport.listen-backlog 128
   ```

Note

The ***transport.listen-backlog*** parameter is not the only parameter that controls queue length. You may also need to change system settings like ***net.ipv4.tcp_max_syn_backlog*** and ***net.core.somaxconn***.

3. Restart all volumes for brick multiplexing to take effect.

```
# gluster volume stop VOLNAME
# gluster volume start VOLNAME
```

Important

Brick compatibility is determined when the volume starts, and depends on volume options shared between bricks. When brick multiplexing is enabled, Red Hat recommends restarting the volume whenever any volume configuration details are changed in order to maintain the compatibility of the bricks grouped under a single process.

21.2.2. Port Range Configuration

By default, for every brick configured on a Red Hat Gluster Storage server node, one process is created and one port is consumed. If you have a large number of bricks configured on a single server, configuring port range lets you control the range of ports allocated by glusterd to newly created or existing bricks and volumes.

This can be achieved with the help of the **glusterd.vol** file. The ***base-port*** and ***max-port*** options can be used to set the port range. By default, ***base-port*** is set to 49152, and ***max-port*** is set to 65535.

Important

If glusterd runs out of free ports to allocate within the specified range of ***base-port*** and ***max-port***, newer bricks and volumes fail to start.

Configuring Port Range

1. Edit the **glusterd.vol** file on all the nodes.

```
# vi /etc/glusterfs/glusterd.vol
```

2. Remove the comment marker # corresponding to the ***base-port*** and ***max-port*** options.

```
volume management
    type mgmt/glusterd
    option working-directory /var/lib/glusterd
    option transport-type socket,rdma
    option transport.socket.keepalive-time 10
    option transport.socket.keepalive-interval 2
    option transport.socket.read-fail-log off
    option ping-timeout 0
    option event-threads 1
#   option lock-timer 180
#   option transport.address-family inet6
    option base-port 49152
    option max-port  65535
end-volume
```

3. Define the port number in the ***base-port***, and ***max-port*** options.

   ```
      option base-port 49152
      option max-port  65535
   ```

4. Save the **glusterd.vol** file and restart the **glusterd** service on each Red Hat Gluster Storage node.

21.3. Network

Data traffic Network becomes a bottleneck as and when number of storage nodes increase. By adding a 10GbE or faster network for data traffic, you can achieve faster per node performance. Jumbo frames must be enabled at all levels, that is, client, Red Hat Gluster Storage node, and ethernet switch levels. MTU of size N+208 must be supported by ethernet switch where N=9000. We recommend you to have a separate network for management and data traffic when protocols like NFS /CIFS are used instead of native client. Preferred bonding mode for Red Hat Gluster Storage client is mode 6 (balance-alb), this allows client to transmit writes in parallel on separate NICs much of the time.

21.4. Memory

Red Hat Gluster Storage does not consume significant compute resources from the storage nodes themselves. However, read intensive workloads can benefit greatly from additional RAM.

21.4.1. Virtual Memory Parameters

The data written by the applications is aggregated in the operating system page cache before being flushed to the disk. The aggregation and writeback of dirty data is governed by the Virtual Memory parameters. The following parameters may have a significant performance impact:

- vm.dirty_ratio
- vm.dirty_background_ratio

The appropriate values of these parameters vary with the type of workload:

- Large-file sequential I/O workloads benefit from higher values for these parameters.
- For small-file and random I/O workloads it is recommended to keep these parameter values low.

The Red Hat Gluster Storage tuned profiles set the values for these parameters appropriately. Hence, it is important to select and activate the appropriate Red Hat Gluster Storage profile based on the workload.

21.5. Small File Performance Enhancements

The ratio of the time taken to perform operations on the metadata of a file to performing operations on its data determines the difference between large files and small files. **`Metadata-intensive workload`** is the term used to identify such workloads. A few performance enhancements can be made to optimize the network and storage performance and minimize the effect of slow throughput and response time for small files in a Red Hat Gluster Storage trusted storage pool.

> **Note**
>
> For a small-file workload, activate the **`rhgs-random-io`** tuned profile.

Configuring Threads for Event Processing

You can set the **`client.event-thread`** and **`server.event-thread`** values for the client and server components. Setting the value to 4, for example, would enable handling four network connections simultaneously.

Setting the event threads value for a client

You can tune the Red Hat Gluster Storage Server performance by tuning the event thread values.

```
# gluster volume set VOLNAME client.event-threads <value>
```

Example 21.1. Tuning the event threads for a client accessing a volume

```
# gluster volume set test-vol client.event-threads 4
```

Setting the event thread value for a server

You can tune the Red Hat Gluster Storage Server performance using event thread values.

```
# gluster volume set VOLNAME server.event-threads <value>
```

Example 21.2. Tuning the event threads for a server accessing a volume

```
# gluster volume set test-vol server.event-threads 4
```

Verifying the event thread values

You can verify the event thread values that are set for the client and server components by executing the following command:

```
# gluster volume info VOLNAME
```

See topic, *Configuring Volume Options* for information on the minimum, maximum, and default values for setting these volume options.

Best practices to tune event threads

It is possible to see performance gains with the Red Hat Gluster Storage stack by tuning the number of threads processing events from network connections. The following are the recommended best practices to tune the event thread values.

1. As each thread processes a connection at a time, having more threads than connections to either the brick processes (**`glusterfsd`**) or the client processes (**`glusterfs`** or **`gfapi`**) is not recommended. Due to this reason, monitor the connection counts (using the **`netstat`** command) on the clients and on the bricks to arrive at an appropriate number for the event thread count.

2. Configuring a higher event threads value than the available processing units could again cause context switches on these threads. As a result reducing the number deduced from the previous step to a number that is less that the available processing units is recommended.

3. If a Red Hat Gluster Storage volume has a high number of brick processes running on a single node, then reducing the event threads number deduced in the previous step would help the competing processes to gain enough concurrency and avoid context switches across the threads.

4. If a specific thread consumes more number of CPU cycles than needed, increasing the event thread count would enhance the performance of the Red Hat Gluster Storage Server.

5. In addition to the deducing the appropriate event-thread count, increasing the **`server.outstanding-rpc-limit`** on the storage nodes can also help to queue the requests for the brick processes and not let the requests idle on the network queue.

6. Another parameter that could improve the performance when tuning the event-threads value is to set the **`performance.io-thread-count`** (and its related thread-counts) to higher values, as these threads perform the actual IO operations on the underlying file system.

21.5.1. Enabling Lookup Optimization

Distribute xlator (DHT) has a performance penalty when it deals with negative lookups. Negative lookups are lookup operations for entries that does not exist in the volume. A lookup for a file/directory that does not exist is a negative lookup.

Negative lookups are expensive and typically slows down file creation, as DHT attempts to find the file in all sub-volumes. This especially impacts small file performance, where a large number of files are being added/created in quick succession to the volume.

The negative lookup fan-out behavior can be optimized by not performing the same in a balanced volume.

The **`cluster.lookup-optimize`** configuration option enables DHT lookup optimization. To enable this option run the following command:

```
# gluster volume set VOLNAME cluster.lookup-optimize <on/off>\
```

Note

The configuration takes effect for newly created directories immediately post setting the above option. For existing directories, a rebalance is required to ensure the volume is in balance before DHT applies the optimization on older directories.

21.6. Replication

If a system is configured for two ways, active-active replication, write throughput will generally be half of what it would be in a non-replicated configuration. However, read throughput is generally improved by replication, as reads can be delivered from either storage node.

21.7. Directory Operations

In order to improve the performance of directory operations of Red Hat Gluster Storage volumes, the maximum metadata (stat, xattr) caching time on the client side is increased to 10 minutes, without compromising on the consistency of the cache.

Significant performance improvements can be achieved in the following workloads by enabling metadata caching:

- Listing of directories (recursive)
- Creating files
- Deleting files
- Renaming files

21.7.1. Enabling Metadata Caching

Enable metadata caching to improve the performance of directory operations. Execute the following commands from any one of the nodes on the trusted storage pool in the order mentioned below.

> **Note**
>
> If majority of the workload is modifying the same set of files and directories simultaneously from multiple clients, then enabling metadata caching might not provide the desired performance improvement.

1. Execute the following command to enable metadata caching and cache invalidation:

   ```
   # gluster volume set <volname> group metadata-cache
   ```

   This is group set option which sets multiple volume options in a single command.

2. To increase the number of files that can be cached, execute the following command:

   ```
   # gluster volume set <VOLNAME> network.inode-lru-limit <n>
   ```

   n, is set to 50000. It can be increased if the number of active files in the volume is very high. Increasing this number increases the memory footprint of the brick processes.

21.8. LVM Cache for Red Hat Gluster Storage

Important

LVM Cache must be used with Red Hat Gluster Storage only on Red Hat Enterprise Linux 7.4 or later. This release includes a number of fixes and enhancements that are critical to a positive experience with caching.

21.8.1. About LVM Cache

An LVM Cache logical volume (LV) can be used to improve the performance of a block device by attaching to it a smaller and much faster device to act as a data acceleration layer. When a cache is attached to an LV, the Linux kernel subsystems attempt to keep 'hot' data copies in the fast cache layer at the block level. Additionally, as space in the cache allows, writes are made initially to the cache layer. The results can be better Input/Output (I/O) performance improvements for many workloads.

21.8.1.1. LVM Cache vs. DM-Cache

`dm-cache` refers to the Linux kernel-level device-mapper subsystem that is responsible for all I/O transactions. For most usual operations, the administrator interfaces with the logical volume manager (LVM) as a much simpler abstraction layer above device-mapper. As such, **`lvmcache`** is simply part of the LVM system acting as an abstraction layer for the **`dm-cache`** subsystem.

21.8.1.2. LVM Cache vs. Gluster Tiered Volumes

Red Hat Gluster Storage supports tiered volumes, which are often configured with the same type of fast devices backing the fast tier bricks. The operation of tiering is at the file level and is distributed across the trusted storage pool (TSP). These tiers operate by moving files between the tiers based on tunable algorithms, such that files are migrated between tiers rather than copied.

In contrast, LVM Cache operates locally at each block device backing the bricks and does so at the block level. LVM Cache stores copies of the hot data in the fast layer using a non-tunable algorithm (though chunk sizes may be tuned for optimal performance).

For most workloads, LVM Cache tends to offer greater performance compared to tiering. However, for certain types of workloads where a large number of clients are consistently accessing the same hot file data set, or where writes can consistently go to the hot tier, tiering may prove more beneficial than LVM Cache.

21.8.1.3. Arbiter Bricks

Arbiter bricks operate by storing all file metadata transactions but not data transactions in order to prevent split-brain problems without the overhead of a third data copy. It is important to understand that file metadata is stored with the file, and so arbiter bricks effectively store empty copies of all files.

In a distributed system such as Red Hat Gluster Storage, latency can greatly affect the performance of file operations, especially when files are very small and file-based transactions are very high. With such small files, the overhead of the metadata latency can be more impactful to performance than the throughput of the I/O subsystems. Therefore, it is important when creating arbiter bricks that the backing storage devices be as fast as the fastest data storage devices. Therefore, when using LVM Cache to accelerate your data volumes with fast devices, you must allocate the same class of fast devices to serve as your arbiter brick backing devices, otherwise your slow arbiter bricks could negate the performance benefits of your cache-accelerated data bricks.

21.8.1.4. Writethrough vs. Writeback

LVM Cache can operate in either writethrough or writeback mode, with writethrough being the default. In writethrough mode, any data written is stored both in the cache layer and in the main data layer. The loss of a device associated with the cache layer in this case would not mean the loss of any data.

Writeback mode delays the writing of data blocks from the cache layer to the main data layer. This mode can increase write performance, but the loss of a device associated with the cache layer can result in lost data locally.

Note

Data resiliency protects from global data loss in the case of a writeback cache device failure under most circumstances, but edge cases could lead to inconsistent data that cannot be automatically healed.

21.8.1.5. Cache-Friendly Workloads

While LVM Cache has been demonstrated to improve performance for Red Hat Gluster Storage under many use cases, the relative effects vary based on the workload. The benefits of block-based caching means that LVM Cache can be efficient for even larger file workloads. However, some workloads may see little-to-no benefit from LVM Cache, and highly-random workloads or those with very large working sets may even experience a performance degradation. It is highly recommended that you understand your workload and test accordingly before making a significant investment in hardware to accelerate your storage workload.

21.8.2. Choosing the Size and Speed of Your Cache Devices

Sizing a cache appropriately to a workload can be a complicated study, particularly in Red Hat Gluster Storage where the cache is local to the bricks rather than global to the volume. In general, you want to understand the size of your working set as a percentage of your total data set and then size your cache layer with some headroom (10-20%) beyond that working set size to allow for efficient flushes and room to cache new writes. Optimally, the entire working set is kept in the cache, and the overall performance you experience is near that of storing your data directly on the fast devices.

When heavily stressed by a working set that is not well-suited for the cache size, you will begin to see a higher percentage of cache misses and your performance will be inconsistent. You may find that as this cache-to-data imbalance increases, a higher percentage of data operations will drop to the speed of the slower data device. From the perspective of a user, this can sometimes be more frustrating than a device that is consistently slow. Understanding and testing your own workload is essential to making an appropriate cache sizing decision.

When choosing your cache devices, always consider high-endurance enterprise-class drives. These are typically tuned to either read or write intensive workloads, so be sure to inspect the hardware performance details when making your selection. Pay close attention to latency alongside IOPS or throughput, as the high transaction activity of a cache will benefit significantly from lower-latency hardware. When possible, select NVMe devices that use the PCI bus directly rather than SATA/SAS devices, as this will additionally benefit latency.

21.8.3. Configuring LVM Cache

A cache pool is created using logical volume manager (LVM) with fast devices as the physical volumes (PVs). The cache pool is then attached to an existing thin pool (TP) or thick logical volume (LV). Once this is done, block-level caching is immediately enabled for the configured LV, and the dm-cache algorithms will work to keep hot copies of data on the cache pool sub-volume.

Warning

Adding or removing cache pools can be done on active volumes, even with mounted filesystems in use. However, there is overhead to the operation and performance impacts will be seen, especially when removing a cache volume in writeback mode, as a full data sync will need to occur. As with any changes to the I/O stack, there is risk of data loss. All changes must be made with the requisite caution.

In the following example commands, we assume the use of a high-performance NVMe PCI device for caching. These devices typically present with device file paths such as **`/dev/nvme0n1`**. A SATA/SAS device will likely present with a device path such as **`/dev/sdb`**. The following example naming has been used:

- Physical Volume (PV) Name: **`/dev/nvme0n1`**
- Volume Group (VG) Name: **`GVG`**
- Thin pool name: **`GTP`**
- Logical Volume (LV) name: **`GLV`**

Note

There are several different ways to configure LVM Cache. Following is the most simple approach applicable to most use cases. For details and further command examples, see **`lvmcache(7)`**.

1. Create a PV for your fast data device.

   ```
   # pvcreate /dev/nvme0n1
   ```

2. Add the fast data PV to the VG that hosts the LV you intend to cache.

   ```
   # vgextend GVG /dev/nvme0n1
   ```

3. Create the cache pool from your fast data device, reserving space required for metadata during the cache conversion process of your LV.

   ```
   # lvcreate --type cache-pool -l 100%FREE -n cpool GVG /dev/nvme0n1
   ```

4. Convert your existing data thin pool LV into a cache LV.

   ```
   # lvconvert --type cache --cachepool GVG/cpool GVG/GTP
   ```

21.8.4. Managing LVM Cache

21.8.4.1. Changing the Mode of an Existing Cache Pool

An existing cache LV can be converted between writethrough and writeback modes with the **`lvchange`** command. For thin LVs, the command must be run against the tdata subvolume.

```
# lvchange --cachemode writeback GVG/GTP_tdata
```

21.8.4.2. Checking Your Configuration

Use the **lsblk** command to view the new virtual block device layout.

```
# lsblk /dev/{sdb,nvme0n1}
NAME                              MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sdb                                 8:16   0   9.1T  0 disk
└─GVG-GTP_tdata_corig 253:9    0   9.1T  0 lvm
  └─GVG-GTP_tdata     253:3    0   9.1T  0 lvm
    └─GVG-GTP-tpool   253:4    0   9.1T  0 lvm
      ├─GVG-GTP       253:5    0   9.1T  0 lvm
      └─GVG-GLV       253:6    0   9.1T  0 lvm  /mnt
nvme0n1                           259:0    0 745.2G  0 disk
├─GVG-GTP_tmeta       253:2    0    76M  0 lvm
│ └─GVG-GTP-tpool     253:4    0   9.1T  0 lvm
│   ├─GVG-GTP         253:5    0   9.1T  0 lvm
│   └─GVG-GLV         253:6    0   9.1T  0 lvm  /mnt
├─GVG-cpool_cdata           253:7    0 701.1G  0 lvm
│ └─GVG-GTP_tdata     253:3    0   9.1T  0 lvm
│   └─GVG-GTP-tpool   253:4    0   9.1T  0 lvm
│     ├─GVG-GTP       253:5    0   9.1T  0 lvm
│     └─GVG-GLV       253:6    0   9.1T  0 lvm  /mnt
├─GVG-cpool_cmeta           253:8    0    48M  0 lvm
│ └─GVG-GTP_tdata     253:3    0   9.1T  0 lvm
│   └─GVG-GTP-tpool   253:4    0   9.1T  0 lvm
│     ├─GVG-GTP       253:5    0   9.1T  0 lvm
│     └─GVG-GLV       253:6    0   9.1T  0 lvm  /mnt
└─GVG-GTP_tdata_corig 253:9    0   9.1T  0 lvm
  └─GVG-GTP_tdata     253:3    0   9.1T  0 lvm
    └─GVG-GTP-tpool   253:4    0   9.1T  0 lvm
      ├─GVG-GTP       253:5    0   9.1T  0 lvm
      └─GVG-GLV       253:6    0   9.1T  0 lvm  /mnt
```

The **lvs** command displays a number of valuable columns to show the status of your cache pool and volume. For more details, see **lvs(8)**.

```
# lvs -a -o name,vg_name,size,pool_lv,devices,cachemode,chunksize
LV                      VG          LSize  Pool      Devices
CacheMode     Chunk
GLV                 GVG    9.10t GTP
0
GTP                 GVG   <9.12t           GTP_tdata(0)
8.00m
[GTP_tdata]         GVG   <9.12t [cpool]   GTP_tdata_corig(0) writethrough
736.00k
[GTP_tdata_corig] GVG   <9.12t           /dev/sdb(0)
0
[GTP_tdata_corig] GVG   <9.12t           /dev/nvme0n1(185076)
0
[GTP_tmeta]         GVG   76.00m           /dev/nvme0n1(185057)
0
[cpool]                 GVG <701.10g           cpool_cdata(0)
writethrough 736.00k
[cpool_cdata]           GVG <701.10g           /dev/nvme0n1(24)
0
[cpool_cmeta]           GVG   48.00m           /dev/nvme0n1(12)
```

```
0
[lvol0_pmspare]         GVG   76.00m           /dev/nvme0n1(0)
0
[lvol0_pmspare]         GVG   76.00m           /dev/nvme0n1(185050)
0
root                    vg_root     50.00g           /dev/sda3(4095)
0
swap                    vg_root    <16.00g           /dev/sda3(0)
0
```

Some of the useful columns from the **lvs** command that can be used to monitor the effectiveness of the cache and to aid in sizing decisions are:

- CacheTotalBlocks
- CacheUsedBlocks
- CacheDirtyBlocks
- CacheReadHits
- CacheReadMisses
- CacheWriteHits
- CacheWriteMisses

You will see a high ratio of Misses to Hits when the cache is cold (freshly attached to the LV). However, with a warm cache (volume online and transacting data for a sufficiently long period of time), high ratios here are indicative of an undersized cache device.

```
# lvs -a -o devices,cachetotalblocks,cacheusedblocks, \
cachereadhits,cachereadmisses | egrep 'Devices|cdata'

Devices                  CacheTotalBlocks  CacheUsedBlocks    CacheReadHits
CacheReadMisses
cpool_cdata(0)                     998850             2581
1                192
```

21.8.4.3. Detaching a Cache Pool

You can split a cache pool from an LV in one command, leaving the data LV in an un-cached state with all data intact and the cache pool still existing but unattached. In writeback mode this can take a long time to complete while all data is synced. This may also negatively impact performance while it is running.

```
# lvconvert --splitcache GVG/cpool
```

Chapter 22. Nagios Configuration Files

Auto-discovery creates folders and files as part of configuring Red Hat Gluster Storage nodes for monitoring. All nodes in the trusted storage pool are configured as hosts in Nagios. The Host and Hostgroup configurations are also generated for trusted storage pool with cluster name. Ensure that the following files and folders are created with the details described to verify the Nagios configurations generated using Auto-discovery.

- In **`/etc/nagios/gluster/`** directory, a new directory **`Cluster-Name`** is created with the name provided as **`Cluster-Name`** while executing **`configure-gluster-nagios`** command for auto-discovery. All configurations created by auto-discovery for the cluster are added in this folder.

- In **`/etc/nagios/gluster/`*****`Cluster-Name`*** directory, a configuration file, ***`Cluster-Name`*****`.cfg`** is generated. This file has the host and hostgroup configurations for the cluster. This also contains service configuration for all the cluster/volume level services.

  The following Nagios object definitions are generated in ***`Cluster-Name`*****`.cfg`** file:

  - A hostgroup configuration with **`hostgroup_name`** as cluster name.
  - A host configuration with **`host_name`** as cluster name.
  - The following service configurations are generated for cluster monitoring:
    - A *Cluster - Quorum* service to monitor the cluster quorum.
    - A *Cluster Utilization* service to monitor overall utilization of volumes in the cluster. This is created only if there is any volume present in the cluster.
    - A *Cluster Auto Config* service to periodically synchronize the configurations in Nagios with Red Hat Gluster Storage trusted storage pool.
  - The following service configurations are generated for each volume in the trusted storage pool:
    - A Volume Status- *Volume-Name* service to monitor the status of the volume.
    - A Volume Utilization - *Volume-Name* service to monitor the utilization statistics of the volume.
    - A Volume Quota - *Volume-Name* service to monitor the Quota status of the volume, if Quota is enabled for the volume.
    - A Volume Self-Heal - *Volume-Name* service to monitor the Self-Heal status of the volume, if the volume is of type replicate or distributed-replicate.
    - A Volume Geo-Replication - *Volume-Name* service to monitor the Geo Replication status of the volume, if Geo-replication is configured for the volume.

- In **`/etc/nagios/gluster/`*****`Cluster-Name`*** directory, a configuration file with name ***`Host-Name`*****`.cfg`** is generated for each node in the cluster. This file has the host configuration for the node and service configuration for bricks from the particular node. The following Nagios object definitions are generated in ***`Host-name`*****`.cfg`**.

  - A host configuration which has *Cluster-Name* in the **`hostgroups`** field.
  - The following services are created for each brick in the node:
    - A *Brick Utilization - brick-path* service to monitor the utilization of the brick.
    - A *Brick - brick-path* service to monitor the brick status.

Table 22.1. Nagios Configuration Files

File Name	Description
`/etc/nagios/nagios.cfg`	Main Nagios configuration file.
`/etc/nagios/cgi.cfg`	CGI configuration file.
`/etc/httpd/conf.d/nagios.conf`	Nagios configuration for httpd.
`/etc/nagios/passwd`	Password file for Nagios users.
`/etc/nagios/nrpe.cfg`	NRPE configuration file.
`/etc/nagios/gluster/gluster-contacts.cfg`	Email notification configuration file.
`/etc/nagios/gluster/gluster-host-services.cfg`	Services configuration file that's applied to every Red Hat Gluster Storage node.
`/etc/nagios/gluster/gluster-host-groups.cfg`	Host group templates for a Red Hat Gluster Storage trusted storage pool.
`/etc/nagios/gluster/gluster-commands.cfg`	Command definitions file for Red Hat Gluster Storage Monitoring related commands.
`/etc/nagios/gluster/gluster-templates.cfg`	Template definitions for Red Hat Gluster Storage hosts and services.
`/etc/nagios/gluster/snmpmanagers.conf`	SNMP notification configuration file with the IP address and community name of SNMP managers where traps need to be sent.

Part VI. Security

Chapter 23. Configuring Network Encryption in Red Hat Gluster Storage

Network encryption is the process of converting data into a cryptic format or code so that it can be securely transmitted on a network. Encryption prevents unauthorized use of the data.

Red Hat Gluster Storage supports network encryption using TLS/SSL. When network encryption is enabled, Red Hat Gluster Storage uses TLS/SSL for authentication and authorization, in place of the authentication framework that is used for non-encrypted connections. The following types of encryption are supported:

I/O encryption

Encryption of the I/O connections between the Red Hat Gluster Storage clients and servers.

Management encryption

Encryption of management (**`glusterd`**) connections within a trusted storage pool, and between **`glusterd`** and NFS Ganesha or SMB clients.

Network encryption is configured in the following files:

`/etc/ssl/glusterfs.pem`

Certificate file containing the system's uniquely signed TLS certificate. This file is unique for each system and must not be shared with others.

`/etc/ssl/glusterfs.key`

This file contains the system's unique private key. This file must not be shared with others.

`/etc/ssl/glusterfs.ca`

This file contains the certificates of the Certificate Authorities (CA) who have signed the certificates. The **`glusterfs.ca`** file must be identical on all servers in the trusted pool, and must contain the certificates of the signing CA for all servers and all clients. All clients should also have a **`.ca`** file that contains the certificates of the signing CA for all the servers.

Red Hat Gluster Storage does not use the global CA certificates that come with the system, so you need to either create your own self-signed certificates, or create certificates and have them signed by a Certificate Authority. If you are using self-signed certificates, the CA file for the servers is a concatenation of the relevant **`.pem`** files of every server and every client. The client CA file is a concatenation of the certificate files of every server.

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

This file is required for management encryption. It enables encryption on the management (**`glusterd`**) connections between **`glusterd`** of all servers and the connection between clients, and contains any configuration required by the Certificate Authority. The **`glusterd`** service of all servers uses this file to fetch volfiles and notify the clients with the volfile changes. This file must be present on all servers and all clients for management encryption to work correctly. It can be empty, but most configurations require at least one line to set the certificate depth (***`transport.socket.ssl-cert-depth`***) required by the Certificate Authority.

23.1. Preparing Certificates

To configure network encryption, each server and client needs a signed certificate and a private key. There are two options for certificates.

Self-signed certificate

Generating and signing the certificate yourself.

Certificate Authority (CA) signed certificate

Generating the certificate and then requesting that a Certificate Authority sign it.

Both of these options ensure that data transmitted over the network cannot be accessed by a third party, but certificates signed by a Certificate Authority imply an added level of trust and verification to a customer using your storage.

Procedure 23.1. Preparing a self-signed certificate

1. **Generate and sign certificates for each server and client**

   Perform the following steps on each server and client.

   a. **Generate a private key for this machine**

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

   b. **Generate a self-signed certificate for this machine**

   The following command generates a signed certificate that expires in 365 days, instead of the default 30 days. Provide a short name for this machine in place of *COMMONNAME*. This is generally a hostname, FQDN, or IP address.

   ```
   # openssl req -new -x509 -key /etc/ssl/glusterfs.key -subj
   "/CN=COMMONNAME" -days 365 -out /etc/ssl/glusterfs.pem
   ```

2. **Generate client-side certificate authority lists**

   From the first server, concatenate the **`/etc/ssl/glusterfs.pem`** files from all servers into a single file called **`glusterfs.ca`**, and place this file in the **`/etc/ssl`** directory on all clients.

   For example, running the following commands from **`server1`** creates a certificate authority list (**`.ca`** file) that contains the certificates (**`.pem`** files) of two servers, and copies the certificate authority list (**`.ca`** file) to three clients.

   ```
   # cat /etc/ssl/glusterfs.pem > /etc/ssl/glusterfs.ca
   # ssh user@server2 cat /etc/ssl/glusterfs.pem >> /etc/ssl/glusterfs.ca
   # scp /etc/ssl/glusterfs.ca client1:/etc/ssl/glusterfs.ca
   # scp /etc/ssl/glusterfs.ca client2:/etc/ssl/glusterfs.ca
   # scp /etc/ssl/glusterfs.ca client3:/etc/ssl/glusterfs.ca
   ```

3. **Generate server-side `glusterfs.ca` files**

   From the first server, append the certificates (**`/etc/ssl/glusterfs.pem`** files) from all clients to the end of the certificate authority list (**`/etc/ssl/glusterfs.ca`** file) generated in the previous step.

   For example, running the following commands from **`server1`** appends the certificates (**`.pem`** files) of three clients to the certificate authority list (**`.ca`** file) on **`server1`**, and then copies that certificate authority list (**`.ca`** file) to one other server.

```
# ssh user@client1 cat /etc/ssl/glusterfs.pem >> /etc/ssl/glusterfs.ca
# ssh user@client2 cat /etc/ssl/glusterfs.pem >> /etc/ssl/glusterfs.ca
# ssh user@client3 cat /etc/ssl/glusterfs.pem >> /etc/ssl/glusterfs.ca
# scp /etc/ssl/glusterfs.ca server2:/etc/ssl/glusterfs.ca
```

4. **Verify server certificates**

   Run the following command in the **/etc/ssl** directory on the servers to verify the certificate on that machine against the Certificate Authority list.

   ```
   # openssl verify -verbose -CAfile glusterfs.ca glusterfs.pem
   ```

   Your certificate is correct if the output of this command is **glusterfs.pem: OK**.

   Note

   This process does not work for self-signed client certificates.

Procedure 23.2. Preparing a Common Certificate Authority certificate

Perform the following steps on each server and client you wish to authorize.

1. **Generate a private key**

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

2. **Generate a certificate signing request**

   The following command generates a certificate signing request for a certificate that expires in 365 days, instead of the default 30 days. Provide a short name for this machine in place of *COMMONNAME*. This is generally a hostname, FQDN, or IP address.

   ```
   # openssl req -new -sha256 -key /etc/ssl/glusterfs.key -subj '/CN=
   <COMMONNAME>' -days 365 -out glusterfs.csr
   ```

3. **Send the generated glusterfs.csr file to your Certificate Authority**

   Your Certificate Authority provides a signed certificate for this machine in the form of a **.pem** file, and the certificates of the Certificate Authority in the form of a **.ca** file.

4. **Place the .pem file provided by the Certificate Authority**

   Ensure that the **.pem** file is called **glusterfs.pem**. Place this file in the **/etc/ssl** directory of this server only.

5. **Place the .ca file provided by the Certificate Authority**

   Ensure that the **.ca** file is called **glusterfs.ca**. Place the **.ca** file in the **/etc/ssl** directory of all servers.

6. **Verify your certificates**

Run the following command in the **/etc/ssl** directory on all clients and servers to verify the certificate on that machine against the Certificate Authority list.

```
# openssl verify -verbose -CAfile glusterfs.ca glusterfs.pem
```

Your certificate is correct if the output of this command is **glusterfs.pem: OK**.

23.2. Configuring Network Encryption for a New Trusted Storage Pool

Follow this section to configure I/O and management encryption on a freshly installed Red Hat Gluster Storage deployment that does not yet have a trusted storage pool configured.

23.2.1. Enabling Management Encryption

Red Hat recommends enabling both management and I/O encryption, but if you only want to use I/O encryption, you can skip this section and continue with Section 23.2.2, “Enabling I/O Encryption”.

Procedure 23.3. Enabling management encryption on servers

Perform the following steps on all servers.

1. **Create and edit the secure-access file**

   Create a new **/var/lib/glusterd/secure-access** file. This file can be empty if you are using the default settings.

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

   Your Certificate Authority may require changes to the SSL certificate depth setting, ***transport.socket.ssl-cert-depth***, in order to work correctly. To edit this setting, add the following line to the **secure-access** file, replacing *n* with the certificate depth required by your Certificate Authority.

   ```
   echo "option transport.socket.ssl-cert-depth n" >
   /var/lib/glusterd/secure-access
   ```

2. **Start glusterd**

   On Red Hat Enterprise Linux 7 based servers, run:

   ```
   # systemctl start glusterd
   ```

   On Red Hat Enterprise Linux 6 based servers, run:

   ```
   # service glusterd start
   ```

3. **Continue storage configuration**

   Proceed with the normal configuration process by setting up the trusted storage pool, formatting bricks, and creating volumes. For more information, see Chapter 4, *Adding Servers to the Trusted Storage Pool* and Chapter 5, *Setting Up Storage Volumes*.

Procedure 23.4. Enabling management encryption on clients

Prerequisites

- You must have configured a trusted storage pool, bricks, and volumes before following this process. For more information, see Chapter 4, *Adding Servers to the Trusted Storage Pool* and Chapter 5, *Setting Up Storage Volumes*.

Perform the following steps on all clients.

1. **Create and edit the secure-access file**

   Create the **`/var/lib/glusterd`** directory, and create a new **`/var/lib/glusterd/secure-access`** file. This file can be empty if you are using the default settings.

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

   Your Certificate Authority may require changes to the SSL certificate depth setting, ***`transport.socket.ssl-cert-depth`***, in order to work correctly. To edit this setting, add the following line to the **`secure-access`** file, replacing *n* with the certificate depth required by your Certificate Authority.

   ```
   echo "option transport.socket.ssl-cert-depth n" >
   /var/lib/glusterd/secure-access
   ```

2. **Start the volume**

   On the server, start the volume.

   ```
   # gluster volume start volname
   ```

3. **Mount the volume**

   The process for mounting a volume depends on the protocol your client is using. The following command mounts a volume called **`testvol`** using the native FUSE protocol.

   ```
   # mount -t glusterfs server1:testvol /mnt/glusterfs
   ```

23.2.2. Enabling I/O Encryption

Follow this section to enable I/O encryption between servers and clients.

Procedure 23.5. Enabling I/O encryption

Prerequisites

- You must have volumes configured, but not started, to perform this process. See Chapter 5, *Setting Up Storage Volumes* for information on creating volumes. To stop a volume, run the following command:

  ```
  # gluster volume stop volname
  ```

Run the following commands from any Gluster server.

1. **Specify servers and clients to allow**

   Provide a list of the common names of servers and clients that are allowed to access the volume. The common names provided must be exactly the same as the common name specified when you created the **glusterfs.pem** file for that server or client.

   ```
   # gluster volume set volname auth.ssl-allow
   'server1,server2,client1,client2,client3'
   ```

   This provides an additional check in case you want to leave keys in place, but temporarily restrict a client or server by removing it from this list, as shown in Section 23.7, “Deauthorizing a Client”.

   You can also use the default value of *****, which indicates that any TLS authenticated machine can mount and access the volume.

2. **Enable TLS/SSL on the volume**

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

3. **Start the volume**

   ```
   # gluster volume start volname
   ```

4. **Verify**

   Verify that the volume can be mounted on authorized clients, and that the volume cannot be mounted by unauthorized clients. The process for mounting a volume depends on the protocol your client is using.

   The process for mounting a volume depends on the protocol your client is using. The following command mounts a volume called **testvol** using the native FUSE protocol.

   ```
   # mount -t glusterfs server1:testvol /mnt/glusterfs
   ```

23.3. Configuring Network Encryption for an existing Trusted Storage Pool

Follow this section to configure I/O and management encryption for an existing Red Hat Gluster Storage Trusted Storage Pool.

23.3.1. Enabling I/O Encryption

Follow this section to enable I/O encryption between servers and clients.

Procedure 23.6. Enabling I/O encryption

1. **Unmount the volume from all clients**

   Unmount the volume by running the following command on all clients.

   ```
   # umount mountpoint
   ```

2. **Stop the volume**

   Stop the volume by running the following command from any server.

   ```
   # gluster volume stop VOLNAME
   ```

3. **Specify servers and clients to allow**

   Provide a list of the common names of servers and clients that are allowed to access the volume. The common names provided must be exactly the same as the common name specified when you created the **glusterfs.pem** file for that server or client.

   ```
   # gluster volume set volname auth.ssl-allow
   'server1,server2,client1,client2,client3'
   ```

   This provides an additional check in case you want to leave keys in place, but temporarily restrict a client or server by removing it from this list, as shown in Section 23.7, “Deauthorizing a Client”.

   You can also use the default value of *, which indicates that any TLS authenticated machine can mount and access the volume.

4. **Enable TLS/SSL encryption on the volume**

   Run the following command from any server to enable TLS/SSL encryption.

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

5. **Start the volume**

   ```
   # gluster volume start volname
   ```

6. **Verify**

   Verify that the volume can be mounted only on authorized clients. The process for mounting a volume depends on the protocol your client is using.

   The following command mounts a volume using the native FUSE protocol. Ensure that this command works on authorized clients, and does not work on unauthorized clients.

   ```
   # mount -t glusterfs server1:/testvolume /mnt/glusterfs
   ```

23.4. Enabling Management Encryption

Red Hat recommends enabling both management and I/O encryption, but if you only want to use I/O encryption, you can skip this section and continue with Section 23.3.1, “Enabling I/O Encryption”.

Prerequisites

- Enabling management encryption requires that storage servers are offline. Schedule an outage window for volumes, applications, clients, and other end users before beginning this process. Be aware that features such as snapshots and geo-replication may also be affected by this outage.

Procedure 23.7. Enabling management encryption

1. **Prepare to enable encryption**

   a. **Unmount all volumes from all clients**

   Run the following command on each client, for each volume mounted on that client.

   ```
   # umount mount-point
   ```

   b. **Stop NFS Ganesha or SMB services, if used**

   Run the following command on any gluster server to disable NFS-Ganesha.

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

   Run the following command on any gluster server to stop SMB.

   ```
   # systemctl stop ctdb
   ```

   c. **Unmount shared storage, if used**

   Run the following command on all servers to unmount shared storage.

   ```
   # umount /var/run/gluster/shared_storage
   ```

   Important

   Features that require shared storage, such as snapshots and geo-replication, may not work until after this process is complete.

   d. **Stop all volumes**

   Run the following command on any server to stop all volumes, including the shared storage volume.

   ```
   # for vol in `gluster volume list`; do gluster --mode=script
   volume stop $vol; sleep 2s; done
   ```

   e. **Stop gluster services on all servers**

   For Red Hat Enterprise Linux 7 based installations:

   ```
   # systemctl stop glusterd
   # pkill glusterfs
   ```

   For Red Hat Enterprise Linux 6 based installations:

   ```
   # service glusterd stop
   # pkill glusterfs
   ```

> **Important**
>
> Bug 1635071 may cause glusterd to crash during shutdown, but there is no functionality impact to this crash. See Resolving glusterd crash for details.

2. **Create and edit the secure-access file on all servers and clients**

   Create a new **`/var/lib/glusterd/secure-access`** file. This file can be empty if you are using the default settings.

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

   Your Certificate Authority may require changes to the SSL certificate depth setting, ***`transport.socket.ssl-cert-depth`***, in order to work correctly. To edit this setting, add the following line to the **`secure-access`** file, replacing *n* with the certificate depth required by your Certificate Authority.

   ```
   echo "option transport.socket.ssl-cert-depth n" >
   /var/lib/glusterd/secure-access
   ```

3. **Clean up after configuring management encryption**

   a. **Start the glusterd service on all servers**

      For Red Hat Enterprise Linux 7 based installations:

      ```
      # systemctl start glusterd
      ```

      For Red Hat Enterprise Linux 6 based installations:

      ```
      # service glusterd start
      ```

   b. **Start all volumes**

      Run the following command on any host to start all volumes including shared storage.

      ```
      # for vol in `gluster volume list`; do gluster --mode=script
      volume start $vol; sleep 2s; done
      ```

   c. **Mount shared storage, if used**

      Run the following command on all servers to mount shared storage.

      ```
      # mount -t glusterfs hostname:/gluster_shared_storage
      /run/gluster/shared_storage
      ```

   d. **Restart NFS Ganesha or SMB services, if used**

      Run the following command on any gluster server to start NFS-Ganesha.

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

Run the following command on any gluster server to start SMB.

```
# systemctl start ctdb
```

e. **Mount volumes on clients**

The process for mounting a volume depends on the protocol your client is using. The following command mounts a volume using the native FUSE protocol.

```
# mount -t glusterfs server1:/testvolume /mnt/glusterfs
```

23.5. Expanding Volumes

Follow this section to add new nodes to a trusted storage pool that uses network encryption.

23.5.1. Certificate signed by a Common Certificate Authority

Follow this section to add a new Gluster server to a trusted storage pool that uses network encryption signed by a common Certificate Authority.

Prerequisites

- Ensure that you have followed the steps in Section 23.1, “Preparing Certificates” before following this section.

Procedure 23.8. Expanding a pool that uses common Certificate Authority signed certificates

1. **Import the common Certificate Authority list**

   Copy the **`/etc/ssl/glusterfs.ca`** file from an existing server into the **`/etc/ssl`** directory of the new server.

2. **For management encryption, create and edit the secure-access file**

   Create a new **`/var/lib/glusterd/secure-access`** file. This file can be empty if you are using the default settings.

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

   Your Certificate Authority may require changes to the SSL certificate depth setting, ***`transport.socket.ssl-cert-depth`***, in order to work correctly. To edit this setting, add the following line to the **`secure-access`** file, replacing *n* with the certificate depth required by your Certificate Authority.

   ```
   echo "option transport.socket.ssl-cert-depth n" >
   /var/lib/glusterd/secure-access
   ```

3. **Start glusterd on the new server**

   ```
   # systemctl start glusterd
   ```

4. **Specify servers and clients to allow**

   Provide a list of the common names of servers and clients that are allowed to access the volume. The common names provided must be exactly the same as the common name specified when you created the **`glusterfs.pem`** file for that server or client.

   ```
   # gluster volume set volname auth.ssl-allow
   'server1,server2,client1,client2,client3'
   ```

   This provides an additional check in case you want to leave keys in place, but temporarily restrict a client or server by removing it from this list, as shown in Section 23.7, “Deauthorizing a Client”.

   > **Note**
   >
   > The **`gluster volume set`** command does not append to existing values of the options. To append the new name to the list, get the existing list using **`gluster volume info`** command, append the new name to the list and set the option again using **`gluster volume set`** command.

   You can also use the default value of **`*`**, which indicates that any TLS authenticated machine can mount and access the volume.

5. **Expand volumes to the new server**

   Follow the instructions in Section 11.7, “Expanding Volumes” to expand existing volumes using the newly trusted server.

23.5.2. Self-signed Certificates

Prerequisites

- Because self-signed certificates are not automatically generated and updated, the trusted storage pool must be offline for this process. Schedule an outage window for volumes, applications, clients, and other end users before beginning this process.

Procedure 23.9. Expanding a pool that uses self-signed certificates

1. **Generate the key and self-signed certificate for the new server**

   Follow the steps in Section 23.1, “Preparing Certificates” to generate a private key and a self-signed certificate for the new server.

2. **Update server Certificate Authority list files**

   Append the contents of the new server's **`/etc/ssl/glusterfs.pem`** file to the **`/etc/ssl/glusterfs.ca`** file on all existing servers in the trusted storage pool.

3. **Update client Certificate Authority list files**

   Append the contents of the new server's **`/etc/ssl/glusterfs.pem`** file to the **`/etc/ssl/glusterfs.ca`** file on all authorized clients in the trusted storage pool.

4. **Stop all gluster processes**

Run the following commands on all servers.

```
# systemctl stop glusterd
# pkill glusterfs
```

Important

Bug 1635071 may cause glusterd to crash during shutdown, but there is no functionality impact to this crash. See Resolving glusterd crash for details.

5. **(Optional) Enable management encryption on the new server**

   Copy the **`/var/lib/glusterd/secure-access`** file from an existing server to the new server.

6. **Start glusterd on the new server**

   ```
   # systemctl start glusterd
   ```

7. **Update servers and clients to allow**

   Run the following command from any server to specify the common names of servers and clients that are allowed to access the volume. The common names provided must be exactly the same as the common name specified when you created the **`glusterfs.pem`** file for that server or client.

   ```
   # gluster volume set volname auth.ssl-allow
   'server1,server2,client1,client2,client3'
   ```

   Note

   The **`gluster volume set`** command does not append to existing values of the options. To append the new name to the list, get the existing list using **`gluster volume info`** command, append the new name to the list and set the option again using **`gluster volume set`** command.

   You can also use the default value of **`*`**, which indicates that any TLS authenticated machine can mount and access the volume.

8. **Restart the glusterfs processes on existing servers and clients**

   a. **On all clients, unmount all volumes**

      ```
      # umount mountpoint
      ```

   b. **On any server, stop all volumes**

      ```
      # for vol in `gluster volume list`; do gluster --mode=script
      volume stop $vol; sleep 2s; done
      ```

c. **On all servers, restart glusterd**

For Red Hat Enterprise Linux 7 based installations:

```
# systemctl start glusterd
```

For Red Hat Enterprise Linux 6 based installations:

```
# service glusterd start
```

d. **On any server, start all volumes**

```
# gluster volume start volname
```

9. **Mount the volume on all clients**

   The process for mounting a volume depends on the protocol your client is using. The following command mounts a volume using the native FUSE protocol.

   ```
   # mount -t glusterfs server1:/test-volume /mnt/glusterfs
   ```

10. **Expand volumes to the new server**

    Follow the instructions in Section 11.7, “Expanding Volumes” to expand existing volumes using the newly trusted server.

23.6. Authorizing a New Client

Follow this section to allow a new client to access a storage pool that uses network encryption.

23.6.1. Certificate Signed by a Common Certificate Authority

Follow this section to authorize a new client to access a trusted storage pool that uses network encryption signed by a common Certificate Authority.

Procedure 23.10. Authorizing a new client using a CA-signed certificate

1. **Generate a key for the client**

   Run the following command on the client.

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

2. **Generate a certificate signing request**

   The following command generates a certificate signing request for a certificate that expires in 365 days, instead of the default 30 days. Provide a short name for this machine in place of *COMMONNAME*. This is generally a hostname, FQDN, or IP address.

   ```
   # openssl req -new -sha256 -key /etc/ssl/glusterfs.key -subj '/CN=
   <COMMONNAME>' -days 365 -out glusterfs.csr
   ```

3. **Send the generated glusterfs.csr file to your Certificate Authority**

   Your Certificate Authority provides a signed certificate for this machine in the form of a **`.pem`** file, and the Certificate Authority list in the form of a **`.ca`** file.

4. **Add provided certificate file on the client**

   Place the **`.pem`** file provided by the Certificate Authority in the **`/etc/ssl`** directory on the client. Ensure that the **`.pem`** file is called **`glusterfs.pem`**.

5. **Add the Certificate Authority list to the client**

   Copy the **`/etc/ssl/glusterfs.ca`** file from an existing client to your new client.

   ```
   # scp existingclient/etc/ssl/glusterfs.ca
   newclient:/etc/ssl/glusterfs.ca
   ```

6. **Verify your certificate**

   Run the following command in the **`/etc/ssl`** directory to verify the certificate on that machine against the Certificate Authority list.

   ```
   # openssl verify -verbose -CAfile glusterfs.ca glusterfs.pem
   ```

   Your certificate is correct if the output of this command is **`glusterfs.pem: OK`**.

7. **Configure management encryption, if used**

   On the client, create the **`/var/lib/glusterd`** directory, and create a new **`/var/lib/glusterd/secure-access`** file. This file can be empty if you are using the default settings.

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

   Your Certificate Authority may require changes to the SSL certificate depth setting, ***`transport.socket.ssl-cert-depth`***, in order to work correctly. To edit this setting, add the following line to the **`secure-access`** file, replacing *n* with the certificate depth required by your Certificate Authority.

   ```
   echo "option transport.socket.ssl-cert-depth n" >
   /var/lib/glusterd/secure-access
   ```

8. **Update the list of servers and clients to allow**

   Run the following command from any server to specify the common names of servers and clients that are allowed to access the volume. The common names provided must be exactly the same as the common name specified when you created the **`glusterfs.pem`** file for that server or client.

   ```
   # gluster volume set volname auth.ssl-allow
   'server1,server2,client1,client2,client3'
   ```

Note

The **`gluster volume set`** command does not append to existing values of the options. To append the new name to the list, get the existing list using **`gluster volume info`** command, append the new name to the list and set the option again using **`gluster volume set`** command.

You can also use the default value of **`*`**, which indicates that any TLS authenticated machine can mount and access the volume.

9. **Start the volume**

```
# gluster volume start volname
```

10. **Verify**

Verify that the volume can be mounted from the new client. The process for mounting a volume depends on the protocol your client is using.

The following command mounts a volume using the native FUSE protocol. Ensure that this command works on authorized clients, and does not work on unauthorized clients.

```
# mount -t glusterfs server1:testvolume /mnt/glusterfs
```

23.6.2. Self-signed Certificates

Prerequisites

- Because self-signed certificates are not automatically generated and updated, the trusted storage pool must be offline for this process. Schedule an outage window for volumes, applications, clients, and other end users before beginning this process.

Follow this section to authorize a new client to access a trusted storage pool that uses network encryption with self-signed certificates.

Procedure 23.11. Authorizing a new client using a self-signed certificate

1. **Generate a key for the client**

Run the following command on the client.

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

2. **Generate a self-signed certificate for the client**

The following command generates a signed certificate that expires in 365 days, instead of the default 30 days. Provide a short name for this machine in place of *COMMONNAME*. This is generally a hostname, FQDN, or IP address.

```
# openssl req -new -x509 -key /etc/ssl/glusterfs.key -subj
"/CN=COMMONNAME" -days 365 -out /etc/ssl/glusterfs.pem
```

3. **Add the Certificate Authority list to the client**

   Copy the **/etc/ssl/glusterfs.ca** file from an existing client to your new client. Run the following command from the new client.

   ```
   # scp existingclient:/etc/ssl/glusterfs.ca /etc/ssl/glusterfs.ca
   ```

4. **Generate new server `glusterfs.ca` files**

   On any server, append the value of the new client's **/etc/ssl/glusterfs.pem** file to the end of the server's **/etc/ssl/glusterfs.ca** file.

   Place the updated **/etc/ssl/glusterfs.ca** file in the **/etc/ssl** directory of all servers in the trusted storage pool.

   For example, running the following commands on any server updates the **glusterfs.ca** file with the **.pem** file from the new client, and then copies that **glusterfs.ca** file to all servers.

   ```
   # ssh user@newclient cat /etc/ssl/glusterfs.pem >>
   /etc/ssl/glusterfs.ca
   # scp /etc/ssl/glusterfs.ca server1:/etc/ssl/glusterfs.ca
   # scp /etc/ssl/glusterfs.ca server2:/etc/ssl/glusterfs.ca
   ```

5. **Configure management encryption on the new client, if used**

   On the client, create the **/var/lib/glusterd** directory, and create a new **/var/lib/glusterd/secure-access** file. This file can be empty if you are using the default settings.

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

   Your Certificate Authority may require changes to the SSL certificate depth setting, ***transport.socket.ssl-cert-depth***, in order to work correctly. To edit this setting, add the following line to the **secure-access** file, replacing *n* with the certificate depth required by your Certificate Authority.

   ```
   echo "option transport.socket.ssl-cert-depth n" >
   /var/lib/glusterd/secure-access
   ```

6. **Update the list of servers and clients to allow**

   Run the following command from any server to specify the common names of servers and clients that are allowed to access the volume. The common names provided must be exactly the same as the common name specified when you created the **glusterfs.pem** file for that server or client.

   ```
   # gluster volume set volname auth.ssl-allow
   'server1,server2,client1,client2,client3'
   ```

> **Note**
>
> The **`gluster volume set`** command does not append to existing values of the options. To append the new name to the list, get the existing list using **`gluster volume info`** command, append the new name to the list and set the option again using **`gluster volume set`** command.

You can also use the default value of `*`, which indicates that any TLS authenticated machine can mount and access the volume.

7. **Start the volume**

   Run the following command from any server to start the volume.

   ```
   # gluster volume start volname
   ```

8. **If management encryption is used, restart glusterd on all servers**

   For Red Hat Enterprise Linux 7 based installations:

   ```
   # systemctl start glusterd
   ```

   For Red Hat Enterprise Linux 6 based installations:

   ```
   # service glusterd start
   ```

9. **Verify**

   Verify that the volume can be mounted from the new client. The process for mounting a volume depends on the protocol your client is using.

   The following command mounts a volume using the native FUSE protocol. Ensure that this command works on authorized clients, and does not work on unauthorized clients.

   ```
   # mount -t glusterfs server1:testvolume /mnt/glusterfs
   ```

23.7. Deauthorizing a Client

To revoke the authorization of a client to access the Red Hat Gluster Storage trusted storage pool, you can do any of the following:

- Remove an authorized client from the allowed list
- Revoke SSL/TLS certificate authorization through a certificate revocation list (CRL)

23.7.1. To Remove an Authorized Client From the Allowed List

Procedure 23.12. Removing an authorized client from the allowed list

1. **List currently authorized clients and servers**

```
$ gluster volume get VOLNAME auth.ssl-allow
```

For example, the following command shows that there are three authorized servers and five authorized clients.

```
$ gluster volume get sample_volname auth.ssl-allow
server1,server2,server3,client1,client2,client3,client4,client5
```

2. **Remove clients to deauthorize from the output**

   For example, if you want to deauthorize client2 and client4, copy the string and remove those clients from the list.

   ```
   server1,server2,server3,client1,client3,client5
   ```

3. **Set the new list of authorized clients and servers**

   Set the value of **`auth.ssl-allow`** to your updated string.

   ```
   $ gluster volume set VOLNAME auth.ssl-allow <list_of_systems>
   ```

   For example, the updated list shows three servers and three clients.

   ```
   $ gluster volume set sample_volname auth.ssl-allow
   server1,server2,server3,client1,client3,client5
   ```

23.7.2. To Revoke SSL/TLS Certificate Authorization Using a SSL Certificate Revocation List

To protect the cluster from malicious or unauthorized network entities, you can specify a path to a file containing SSL certificate revocation list. This path containing the list of revoked certificates enables server nodes to stop the nodes with revoked certificates from accessing the cluster.

For example, you can provide the name of the file containing certifcate revocation list with the **`volume-set`** command as follows:

```
$ gluster volume set vm-images ssl.crl-path /etc/ssl/vm-images-crl.pem
```

Note

Only the CA signed certificates can be revoked and not the self-signed certificates

23.8. Disabling Network Encryption

Follow this section to disable network encryption on clients and servers.

Procedure 23.13. Disabling I/O encryption

1. **Unmount volumes from all clients**

Run the following command on each client for any volume that should have encryption disabled.

```
# umount /mountpoint
```

2. **Stop encrypted volumes**

   Run the following command on any server to stop volumes that should have encryption disabled.

   ```
   # gluster volume stop volname
   ```

3. **Disable server and client SSL usage**

   Run the following commands for each volume that should have encryption disabled.

   ```
   # gluster volume set volname server.ssl off
   # gluster volume set volname client.ssl off
   ```

4. **Start volumes**

   ```
   # gluster volume start volname
   ```

5. **Mount volumes on clients**

   The process for mounting a volume depends on the protocol your client is using. The following command mounts a volume using the native FUSE protocol.

   ```
   # mount -t glusterfs server1:/testvolume /mnt/glusterfs
   ```

Procedure 23.14. Disabling management encryption

1. **Unmount volumes from all clients**

   Run the following command on each client for any volume that should have encryption disabled.

   ```
   # umount /mountpoint
   ```

2. **Stop glusterd on all nodes**

   For Red Hat Enterprise Linux 7 based installations:

   ```
   # systemctl stop glusterd
   ```

   For Red Hat Enterprise Linux 6 based installations:

   ```
   # service glusterd stop
   ```

Important

Bug 1635071 may cause glusterd to crash during shutdown, but there is no functionality impact to this crash. See Resolving glusterd crash for details.

3. **Remove the secure-access file**

   Run the following command on all servers and clients to remove the secure-access file. You can just rename the file if you are only disabling encryption temporarily.

   ```
   # rm -f /var/lib/glusterd/secure-access
   ```

4. **Start glusterd on all nodes**

   For Red Hat Enterprise Linux 7 based installations:

   ```
   # systemctl start glusterd
   ```

   For Red Hat Enterprise Linux 6 based installations:

   ```
   # service glusterd start
   ```

5. **Mount volumes on clients**

   The process for mounting a volume depends on the protocol your client is using. The following command mounts a volume using the native FUSE protocol.

   ```
   # mount -t glusterfs server1:/testvolume /mnt/glusterfs
   ```

Important

If you are permanently disabling network encryption, you can now delete the SSL certificate files. Do not delete these files if you are only disabling encryption temporarily.

Part VII. Troubleshoot

Chapter 24. Resolving Common Issues

This chapter provides some of the Red Hat Gluster Storage troubleshooting methods.

24.1. Identifying locked file and clear locks

You can use the **statedump** command to list the locks held on files. The **statedump** output also provides information on each lock with its range, basename, and PID of the application holding the lock, and so on. You can analyze the output to find the locks whose owner/application is no longer running or interested in that lock. After ensuring that no application is using the file, you can clear the lock using the following **clear-locks** command:

```
# gluster volume clear-locks VOLNAME path kind {blocked | granted | all}{inode
range | entry basename | posix range}
```

For more information on performing **statedump**, see Section 19.8, “Viewing complete volume state with statedump”

To identify locked file and clear locks

1. Perform **statedump** on the volume to view the files that are locked using the following command:

   ```
   # gluster volume statedump VOLNAME
   ```

   For example, to display **statedump** of test-volume:

   ```
   # gluster volume statedump test-volume
   Volume statedump successful
   ```

   The **statedump** files are created on the brick servers in the **/tmp** directory or in the directory set using the **server.statedump-path** volume option. The naming convention of the dump file is ***brick-path*.*brick-pid*.dump**.

2. Clear the entry lock using the following command:

   ```
   # gluster volume clear-locks VOLNAME path kind granted entry basename
   ```

   The following are the sample contents of the **statedump** file indicating entry lock (entrylk). Ensure that those are stale locks and no resources own them.

   ```
   [xlator.features.locks.vol-locks.inode]
   path=/
   mandatory=0
   entrylk-count=1
   lock-dump.domain.domain=vol-replicate-0
   xlator.feature.locks.lock-dump.domain.entrylk.entrylk[0]
   (ACTIVE)=type=ENTRYLK_WRLCK on basename=file1, pid = 714782904,
   owner=ffffff2a3c7f0000, transport=0x20e0670, , granted at Mon Feb 27
   16:01:01 2012

   conn.2.bound_xl./rhgs/brick1.hashsize=14057
   conn.2.bound_xl./rhgs/brick1.name=/gfs/brick1/inode
   conn.2.bound_xl./rhgs/brick1.lru_limit=16384
   conn.2.bound_xl./rhgs/brick1.active_size=2
   conn.2.bound_xl./rhgs/brick1.lru_size=0
   conn.2.bound_xl./rhgs/brick1.purge_size=0
   ```

For example, to clear the entry lock on **`file1`** of test-volume:

```
# gluster volume clear-locks test-volume / kind granted entry file1
Volume clear-locks successful
test-volume-locks: entry blocked locks=0 granted locks=1
```

3. Clear the inode lock using the following command:

   ```
   # gluster volume clear-locks VOLNAME path kind granted inode range
   ```

   The following are the sample contents of the **`statedump`** file indicating there is an inode lock (inodelk). Ensure that those are stale locks and no resources own them.

   ```
   [conn.2.bound_xl./rhgs/brick1.active.1]
   gfid=538a3d4a-01b0-4d03-9dc9-843cd8704d07
   nlookup=1
   ref=2
   ia_type=1
   [xlator.features.locks.vol-locks.inode]
   path=/file1
   mandatory=0
   inodelk-count=1
   lock-dump.domain.domain=vol-replicate-0
   inodelk.inodelk[0](ACTIVE)=type=WRITE, whence=0, start=0, len=0, pid =
   714787072, owner=00ffff2a3c7f0000, transport=0x20e0670, , granted at
   Mon Feb 27 16:01:01 2012
   ```

   For example, to clear the inode lock on **`file1`** of test-volume:

   ```
   # gluster  volume clear-locks test-volume /file1 kind granted inode
   0,0-0
   Volume clear-locks successful
   test-volume-locks: inode blocked locks=0 granted locks=1
   ```

4. Clear the granted POSIX lock using the following command:

   ```
   # gluster volume clear-locks VOLNAME path kind granted posix range
   ```

   The following are the sample contents of the **`statedump`** file indicating there is a granted POSIX lock. Ensure that those are stale locks and no resources own them.

   ```
   xlator.features.locks.vol1-locks.inode]
   path=/file1
   mandatory=0
   posixlk-count=15
   posixlk.posixlk[0](ACTIVE)=type=WRITE, whence=0, start=8, len=1, pid =
   23848, owner=d824f04c60c3c73c, transport=0x120b370, , blocked at Mon
   Feb 27 16:01:01 2012
   , granted at Mon Feb 27 16:01:01 2012

   posixlk.posixlk[1](ACTIVE)=type=WRITE, whence=0, start=7, len=1, pid =
   1, owner=30404152462d436c-69656e7431, transport=0x11eb4f0, , granted
   at Mon Feb 27 16:01:01 2012

   posixlk.posixlk[2](BLOCKED)=type=WRITE, whence=0, start=8, len=1, pid =
   ```

```
1, owner=30404152462d436c-69656e7431, transport=0x11eb4f0, , blocked
at Mon Feb 27 16:01:01 2012

posixlk.posixlk[3](ACTIVE)=type=WRITE, whence=0, start=6, len=1, pid =
12776, owner=a36bb0aea0258969, transport=0x120a4e0, , granted at Mon
Feb 27 16:01:01 2012
...
```

For example, to clear the granted POSIX lock on **file1** of test-volume:

```
# gluster volume clear-locks test-volume /file1 kind granted posix
0,8-1
Volume clear-locks successful
test-volume-locks: posix blocked locks=0 granted locks=1
test-volume-locks: posix blocked locks=0 granted locks=1
test-volume-locks: posix blocked locks=0 granted locks=1
```

5. Clear the blocked POSIX lock using the following command:

   # gluster volume clear-locks *VOLNAME* *path* kind blocked posix *range*

   The following are the sample contents of the **statedump** file indicating there is a blocked POSIX lock. Ensure that those are stale locks and no resources own them.

```
[xlator.features.locks.vol1-locks.inode]
path=/file1
mandatory=0
posixlk-count=30
posixlk.posixlk[0](ACTIVE)=type=WRITE, whence=0, start=0, len=1, pid =
23848, owner=d824f04c60c3c73c, transport=0x120b370, , blocked at Mon
Feb 27 16:01:01 2012
, granted at Mon Feb 27 16:01:01

posixlk.posixlk[1](BLOCKED)=type=WRITE, whence=0, start=0, len=1, pid =
1, owner=30404146522d436c-69656e7432, transport=0x1206980, , blocked
at Mon Feb 27 16:01:01 2012

posixlk.posixlk[2](BLOCKED)=type=WRITE, whence=0, start=0, len=1, pid =
1, owner=30404146522d436c-69656e7432, transport=0x1206980, , blocked
at Mon Feb 27 16:01:01 2012

posixlk.posixlk[3](BLOCKED)=type=WRITE, whence=0, start=0, len=1, pid =
1, owner=30404146522d436c-69656e7432, transport=0x1206980, , blocked
at Mon Feb 27 16:01:01 2012

posixlk.posixlk[4](BLOCKED)=type=WRITE, whence=0, start=0, len=1, pid =
1, owner=30404146522d436c-69656e7432, transport=0x1206980, , blocked
at Mon Feb 27 16:01:01 2012

...
```

For example, to clear the blocked POSIX lock on **file1** of test-volume:

```
# gluster volume clear-locks test-volume /file1 kind blocked posix
0,0-1
```

```
Volume clear-locks successful
test-volume-locks: posix blocked locks=28 granted locks=0
test-volume-locks: posix blocked locks=1 granted locks=0
No locks cleared.
```

6. Clear all POSIX locks using the following command:

   ```
   # gluster volume clear-locks VOLNAME path kind all posix range
   ```

   The following are the sample contents of the **statedump** file indicating that there are POSIX locks. Ensure that those are stale locks and no resources own them.

   ```
   [xlator.features.locks.vol1-locks.inode]
   path=/file1
   mandatory=0
   posixlk-count=11
   posixlk.posixlk[0](ACTIVE)=type=WRITE, whence=0, start=8, len=1, pid =
   12776, owner=a36bb0aea0258969, transport=0x120a4e0, , blocked at Mon
   Feb 27 16:01:01 2012
   , granted at Mon Feb 27 16:01:01 2012

   posixlk.posixlk[1](ACTIVE)=type=WRITE, whence=0, start=0, len=1, pid =
   12776, owner=a36bb0aea0258969, transport=0x120a4e0, , granted at Mon
   Feb 27 16:01:01 2012

   posixlk.posixlk[2](ACTIVE)=type=WRITE, whence=0, start=7, len=1, pid =
   23848, owner=d824f04c60c3c73c, transport=0x120b370, , granted at Mon
   Feb 27 16:01:01 2012

   posixlk.posixlk[3](ACTIVE)=type=WRITE, whence=0, start=6, len=1, pid =
   1, owner=30404152462d436c-69656e7431, transport=0x11eb4f0, , granted
   at Mon Feb 27 16:01:01 2012

   posixlk.posixlk[4](BLOCKED)=type=WRITE, whence=0, start=8, len=1, pid =
   23848, owner=d824f04c60c3c73c, transport=0x120b370, , blocked at Mon
   Feb 27 16:01:01 2012
   ...
   ```

   For example, to clear all POSIX locks on **file1** of test-volume:

   ```
   # gluster volume clear-locks test-volume /file1 kind all posix 0,0-1
   Volume clear-locks successful
   test-volume-locks: posix blocked locks=1 granted locks=0
   No locks cleared.
   test-volume-locks: posix blocked locks=4 granted locks=1
   ```

You can perform **statedump** on test-volume again to verify that all the above locks are cleared.

24.2. Retrieving File Path from the Gluster Volume

The heal info command lists the GFIDs of the files that needs to be healed. If you want to find the path of the files associated with the GFIDs, use the **getfattr** utility. The **getfattr** utility enables you to locate a file residing on a gluster volume brick. You can retrieve the path of a file even if the filename is unknown.

24.2.1. Retrieving Known File Name

To retrieve a file path when the file name is known, execute the following command in the Fuse mount directory:

```
# getfattr -n trusted.glusterfs.pathinfo -e text
<path_to_fuse_mount/filename>
```

Where,

path_to_fuse_mount: The fuse mount where the gluster volume is mounted.

filename: The name of the file for which the path information is to be retrieved.

For example:

```
# getfattr  -n trusted.glusterfs.pathinfo -e text /mnt/fuse_mnt/File1
getfattr: Removing leading '/' from absolute path names
# file: mnt/fuse_mnt/File1
trusted.glusterfs.pathinfo="(<DISTRIBUTE:testvol-dht> (<REPLICATE:testvol-
replicate-0>
<POSIX(/rhgs/brick1):tuxpad:/rhgs/brick1/File1>
<POSIX(/rhgs/brick2):tuxpad:/rhgs/brick2/File1>))"
```

The command output displays the brick pathinfo under the <POSIX> tag. In this example output, two paths are displayed as the file is replicated twice and resides on a two-way replicated volume.

24.2.2. Retrieving Unknown File Name

You can retrieve the file path of an unknown file using its gfid string. The gfid string is the hyphenated version of the **trusted.gfid** attribute. For example, if the gfid is **80b0b1642ea4478ba4cda9f76c1e6efd**, then the gfid string will be **80b0b164-2ea4-478b-a4cd-a9f76c1e6efd**.

Note

To obtain the gfid of a file, run the following command:

```
# getfattr -d -m. -e hex /path/to/file/on/the/brick
```

24.2.3. Retrieving File Path using gfid String

To retrieve the file path using the gfid string, follow these steps:

1. Fuse mount the volume with the aux-gfid option enabled.

   ```
   # mount -t glusterfs -o aux-gfid-mount hostname:volume-name
   <path_to_fuse_mnt>
   ```

   Where,

   path_to_fuse_mount: The fuse mount where the gluster volume is mounted.

For example:

```
# mount -t glusterfs -o aux-gfid-mount 127.0.0.2:testvol
/mnt/aux_mount
```

2. After mounting the volume, execute the following command

```
# getfattr -n trusted.glusterfs.pathinfo -e text <path-to-fuse-
mnt>/.gfid/<GFID string>
```

Where,

path_to_fuse_mount: The fuse mount where the gluster volume is mounted.

GFID string: The GFID string.

For example:

```
# getfattr -n trusted.glusterfs.pathinfo -e text
/mnt/aux_mount/.gfid/80b0b164-2ea4-478b-a4cd-a9f76c1e6efd
getfattr: Removing leading '/' from absolute path names
# file: mnt/aux_mount/.gfid/80b0b164-2ea4-478b-a4cd-a9f76c1e6efd
trusted.glusterfs.pathinfo="(<DISTRIBUTE:testvol-dht>
(<REPLICATE:testvol-replicate-0>
<POSIX(/rhgs/brick2):tuxpad:/rhgs/brick2/File1>
<POSIX(/rhgs/brick1):tuxpad:/rhgs/brick1/File1>))
```

The command output displays the brick pathinfo under the <POSIX> tag. In this example output, two paths are displayed as the file is replicated twice and resides on a two-way replicated volume.

24.2.4. Controlling Self-heal for Dispersed Volumes

For dispersed volumes, when a node with multiple bricks goes offline and comes back online, self-heal daemon starts healing the bricks. This self-heal can lead to high CPU usage in case of large amounts of data and can affect the ongoing I/O operations, thus, decreasing storage efficiency.

To control the CPU and memory usage of the self-heal daemon, follow these steps:

1. Navigate to the scripts folder using the following command:

```
# cd /usr/share/glusterfs/scripts
```

2. Determine the PID of the self-heal daemon using the following command:

```
# ps -aef | grep glustershd
```

The output will be in the following format:

```
root      1565     1  0 Feb05 ?        00:09:17 /usr/sbin/glusterfs -
s localhost --volfile-id gluster/glustershd -p
/var/run/gluster/glustershd/glustershd.pid -l
/var/log/glusterfs/glustershd.log -S
```

```
/var/run/gluster/ed49b959a0dc9b2185913084e3b2b339.socket --xlator-
option *replicate*.node-uuid=13dbfa1e-ebbf-4cee-a1ac-ca6763903c55
root     16766 14420  0 19:00 pts/0    00:00:00 grep --color=auto
glustershd
```

In this output, **1565** represents the PID of the selfheald service.

3. Execute the **`control-cpu-load`** script using the following command:

```
# sh control-cpu-load.sh
```

4. When the system prompts for the following input, type the PID of the self-heal daemon acquired from the previous step and press **`Enter`**:

```
[root@XX-XX scripts]# sh control-cpu-load.sh
Enter gluster daemon pid for which you want to control CPU.
1565
```

5. When the system prompts for the following input, type **`y`** and press **`Enter`**:

```
If you want to continue the script to attach 1565 with new
cgroup_gluster_1565 cgroup Press (y/n)?
```

In this example, **1565** represents the PID of the **selfheald** service. The PID of the selfheald service can vary from system to system.

6. When the system prompts for the following input, enter the required quota value to be assigned to the self-heal daemon and press **`Enter`**:

```
Creating child cgroup directory 'cgroup_gluster_1565 cgroup' for
glustershd.service.
Enter quota value in range [10,100]:
25
```

In this example, the quota value for the self-heal daemon is set as **`25`**.

Note

The recommended quota value for a self-heal daemon is 25. However, the quota value can be set by the user on a run-time basis.

The system prompts the following notification once the quota value is successfully set:

```
Entered quota value is 25
Setting 25000 to cpu.cfs_quota_us for gluster_cgroup.
Tasks are attached successfully specific to 1565 to
cgroup_gluster_1565.
```

To check the CPU usage for the self-heal daemon, execute the **`top`** command.

Important

Perform this procedure every time the daemon is restarted with the new daemon PID.

24.3. Resolving `glusterd` Crash

`glusterd` crash is observed in the following scenarios:

- **`glusterd`** recieves a Termination Signal or **`SIGTERM`**.
- **`Segmentation fault`** error message when upgrading Red Hat Gluster Storage.
- **`glusterd`** service is being stopped.

Important

There is no functionality impact to these crashes as they occur during the shutdown path of **`glusterd`**.

If the **`glusterd`** crash is persistent in any other scenarios, contact Red Hat Support

24.4. Restarting a dead/failed brick

Note

In case of a Red Hat OpenShift Container Storage converged and independent mode, where brick multiplexing is enabled by default, the volumes for which the failed/dead bricks are multiplexed into a single process need to be force started.

If any bricks associated with a volume are down, start the bricks by executing the following command:

```
# gluster volume start VOLNAME force
```

Part VIII. Appendices

Chapter 25. Starting and Stopping the glusterd service

Using the **`glusterd`** command line, logical storage volumes can be decoupled from physical hardware. Decoupling allows storage volumes to be grown, resized, and shrunk, without application or server downtime.

Regardless of changes made to the underlying hardware, the trusted storage pool is always available while changes to the underlying hardware are made. As storage is added to the trusted storage pool, volumes are rebalanced across the pool to accommodate the added storage capacity.

The **`glusterd`** service is started automatically on all servers in the trusted storage pool. The service can also be manually started and stopped as required.

- Run the following command to start glusterd manually.

  ```
  # service glusterd start
  ```

- Run the following command to stop glusterd manually.

  ```
  # service glusterd stop
  ```

> **Important**
>
> If **`glusterd`** crashes, there is no functionality impact to this crash as it occurs during the shutdown. For more information, see Section 24.3, “Resolving **`glusterd`** Crash”

When a Red Hat Gluster Storage server node that hosts a very large number of bricks or snapshots is upgraded, cluster management commands may become unresponsive as glusterd attempts to start all brick processes concurrently for all bricks and snapshots. If you have more than 250 bricks or snapshots being hosted by a single node, Red Hat recommends deactivating snapshots until upgrade is complete.

Chapter 26. Manually Recovering File Split-brain

This chapter provides steps to manually recover from split-brain.

1. Run the following command to obtain the path of the file that is in split-brain:

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

   From the command output, identify the files for which file operations performed from the client keep failing with Input/Output error.

2. Close the applications that opened split-brain file from the mount point. If you are using a virtual machine, you must power off the machine.

3. Obtain and verify the AFR changelog extended attributes of the file using the **`getfattr`** command. Then identify the type of split-brain to determine which of the bricks contains the 'good copy' of the file.

   ```
   getfattr -d -m . -e hex <file-path-on-brick>
   ```

   For example,

   ```
   # getfattr -d -e hex -m. brick-a/file.txt
   \#file: brick-a/file.txt
   security.selinux=0x726f6f743a6f626a6563745f723a66696c655f743a733000
   trusted.afr.vol-client-2=0x000000000000000000000000
   trusted.afr.vol-client-3=0x000000020000000000000000
   trusted.gfid=0x307a5c9efddd4e7c96e94fd4bcdcbd1b
   ```

   The extended attributes with **`trusted.afr.`*`VOLNAME`*`volname-client-<subvolume-index>`** are used by AFR to maintain changelog of the file. The values of the **`trusted.afr.`*`VOLNAME`*`volname-client-<subvolume-index>`** are calculated by the glusterFS client (FUSE or NFS-server) processes. When the glusterFS client modifies a file or directory, the client contacts each brick and updates the changelog extended attribute according to the response of the brick.

   `subvolume-index` is the **`brick number - 1`** of **`gluster volume info`** ***`VOLNAME`*** output.

   For example,

   ```
   # gluster volume info vol
   Volume Name: vol
   Type: Distributed-Replicate
   Volume ID: 4f2d7849-fbd6-40a2-b346-d13420978a01
   Status: Created
   Number of Bricks: 4 x 2 = 8
   Transport-type: tcp
   Bricks:
   brick1: server1:/rhgs/brick1
   brick2: server1:/rhgs/brick2
   brick3: server1:/rhgs/brick3
   brick4: server1:/rhgs/brick4
   ```

```
brick5: server1:/rhgs/brick5
brick6: server1:/rhgs/brick6
brick7: server1:/rhgs/brick7
brick8: server1:/rhgs/brick8
```

In the example above:

```
Brick              |    Replica set        |    Brick subvolume index
-------------------------------------------------------------------
----------
/rhgs/brick1     |       0               |       0
/rhgs/brick2     |       0               |       1
/rhgs/brick3     |       1               |       2
/rhgs/brick4     |       1               |       3
/rhgs/brick5     |       2               |       4
/rhgs/brick6     |       2               |       5
/rhgs/brick7     |       3               |       6
/rhgs/brick8     |       3               |       7
```
```

Each file in a brick maintains the changelog of itself and that of the files present in all the other bricks in it's replica set as seen by that brick.

In the example volume given above, all files in brick-a will have 2 entries, one for itself and the other for the file present in it's replica pair. The following is the changelog for brick2,

- trusted.afr.vol-client-0=0x000000000000000000000000 - is the changelog for itself (brick1)
- trusted.afr.vol-client-1=0x000000000000000000000000 - changelog for brick2 as seen by brick1

Likewise, all files in brick2 will have the following:

- trusted.afr.vol-client-0=0x000000000000000000000000 - changelog for brick1 as seen by brick2
- trusted.afr.vol-client-1=0x000000000000000000000000 - changelog for itself (brick2)

### Note

These files do not have entries for themselves, only for the other bricks in the replica. For example, **`brick1`** will only have **`trusted.afr.vol-client-1`** set and **`brick2`** will only have **`trusted.afr.vol-client-0`** set. Interpreting the changelog remains same as explained below.

The same can be extended for other replica pairs.

### Interpreting changelog (approximate pending operation count) value

Each extended attribute has a value which is 24 hexa decimal digits. First 8 digits represent changelog of data. Second 8 digits represent changelog of metadata. Last 8 digits represent Changelog of directory entries.

Pictorially representing the same is as follows:
```

```
0x 000003d7 00000001 00000000110
        |      |       |
        |      |        \_ changelog of directory entries
        |       \_ changelog of metadata
         \ _ changelog of data
```

For directories, metadata and entry changelogs are valid. For regular files, data and metadata changelogs are valid. For special files like device files and so on, metadata changelog is valid. When a file split-brain happens it could be either be data split-brain or meta-data split-brain or both.

The following is an example of both data, metadata split-brain on the same file:

```
# getfattr -d -m . -e hex /rhgs/brick?/a
getfattr: Removing leading '/' from absolute path names
\#file: rhgs/brick1/a
trusted.afr.vol-client-0=0x000000000000000000000000
trusted.afr.vol-client-1=0x000003d70000000100000000
trusted.gfid=0x80acdbd886524f6fbefa21fc356fed57
\#file: rhgs/brick2/a
trusted.afr.vol-client-0=0x000003b00000000100000000
trusted.afr.vol-client-1=0x000000000000000000000000
trusted.gfid=0x80acdbd886524f6fbefa21fc356fed57
```

Scrutinize the changelogs

The changelog extended attributes on file **`/rhgs/brick1/a`** are as follows:

- The first 8 digits of **`trusted.afr.vol-client-0 are all zeros (0x00000000................)`**,

  The first 8 digits of **`trusted.afr.vol-client-1`** are not all zeros (0x000003d7................).

  So the changelog on **`/rhgs/brick-a/a`** implies that some data operations succeeded on itself but failed on **`/rhgs/brick2/a`**.

- The second 8 digits of **`trusted.afr.vol-client-0 are all zeros (0x........00000000........)`**, and the second 8 digits of **`trusted.afr.vol-client-1`** are not all zeros (0x........00000001........).

  So the changelog on **`/rhgs/brick1/a`** implies that some metadata operations succeeded on itself but failed on **`/rhgs/brick2/a`**.

The changelog extended attributes on file **`/rhgs/brick2/a`** are as follows:

- The first 8 digits of **`trusted.afr.vol-client-0`** are not all zeros (0x000003b0................).

  The first 8 digits of **`trusted.afr.vol-client-1`** are all zeros (0x00000000................).

  So the changelog on **`/rhgs/brick2/a`** implies that some data operations succeeded on itself but failed on **`/rhgs/brick1/a`**.

- The second 8 digits of **`trusted.afr.vol-client-0`** are not all zeros (0x........00000001........)

  The second 8 digits of **`trusted.afr.vol-client-1`** are all zeros (0x........00000000........).

  So the changelog on **`/rhgs/brick2/a`** implies that some metadata operations succeeded on itself but failed on **`/rhgs/brick1/a`**.

Here, both the copies have data, metadata changes that are not on the other file. Hence, it is both data and metadata split-brain.

Deciding on the correct copy

You must inspect **stat** and **getfattr** output of the files to decide which metadata to retain and contents of the file to decide which data to retain. To continue with the example above, here, we are retaining the data of **/rhgs/brick1/a** and metadata of **/rhgs/brick2/a**.

Resetting the relevant changelogs to resolve the split-brain

Resolving data split-brain

You must change the changelog extended attributes on the files as if some data operations succeeded on **/rhgs/brick1/a** but failed on /rhgs/brick-b/a. But **/rhgs/brick2/a** should **not** have any changelog showing data operations succeeded on **/rhgs/brick2/a** but failed on **/rhgs/brick1/a**. You must reset the data part of the changelog on **trusted.afr.vol-client-0** of **/rhgs/brick2/a**.

Resolving metadata split-brain

You must change the changelog extended attributes on the files as if some metadata operations succeeded on **/rhgs/brick2/a** but failed on **/rhgs/brick1/a**. But **/rhgs/brick1/a** should **not** have any changelog which says some metadata operations succeeded on **/rhgs/brick1/a** but failed on **/rhgs/brick2/a**. You must reset metadata part of the changelog on **trusted.afr.vol-client-1** of **/rhgs/brick1/a**

Run the following commands to reset the extended attributes.

a. On **/rhgs/brick2/a**, for **trusted.afr.vol-client-0** **0x000003b00000000100000000** to **0x000000000000000100000000**, execute the following command:

```
# setfattr -n trusted.afr.vol-client-0 -v
0x000000000000000100000000 /rhgs/brick2/a
```

b. On **/rhgs/brick1/a**, for **trusted.afr.vol-client-1** **0x0000000000000000ffffffff** to **0x000003d70000000000000000**, execute the following command:

```
# setfattr -n trusted.afr.vol-client-1 -v
0x000003d70000000000000000 /rhgs/brick1/a
```

After you reset the extended attributes, the changelogs would look similar to the following:

```
# getfattr -d -m . -e hex /rhgs/brick?/a
getfattr: Removing leading '/' from absolute path names
\#file: rhgs/brick1/a
trusted.afr.vol-client-0=0x000000000000000000000000
trusted.afr.vol-client-1=0x000003d70000000000000000
trusted.gfid=0x80acdbd886524f6fbefa21fc356fed57

\#file: rhgs/brick2/a
trusted.afr.vol-client-0=0x000000000000000100000000
trusted.afr.vol-client-1=0x000000000000000000000000
trusted.gfid=0x80acdbd886524f6fbefa21fc356fed57
```

Resolving Directory entry split-brain

AFR has the ability to conservatively merge different entries in the directories when there is a split-brain on directory. If on one brick directory **`storage`** has entries **`1`**, **`2`** and has entries **`3`**, **`4`** on the other brick then AFR will merge all of the entries in the directory to have **`1, 2, 3, 4`** entries in the same directory. But this may result in deleted files to re-appear in case the split-brain happens because of deletion of files on the directory. Split-brain resolution needs human intervention when there is at least one entry which has same file name but different **`gfid`** in that directory.

For example:

On **`brick-a`** the directory has 2 entries **`file1`** with **`gfid_x`** and **`file2`** . On **`brick-b`** directory has 2 entries **`file1`** with **`gfid_y`** and **`file3`**. Here the gfid's of **`file1`** on the bricks are different. These kinds of directory split-brain needs human intervention to resolve the issue. You must remove either **`file1`** on **`brick-a`** or the **`file1`** on **`brick-b`** to resolve the split-brain.

In addition, the corresponding **`gfid-link`** file must be removed. The **`gfid-link`** files are present in the .**`glusterfs`** directory in the top-level directory of the brick. If the gfid of the file is **`0x307a5c9efddd4e7c96e94fd4bcdcbd1b`** (the trusted.gfid extended attribute received from the **`getfattr`** command earlier), the gfid-link file can be found at **`/rhgs/brick1/.glusterfs/30/7a/307a5c9efddd4e7c96e94fd4bcdcbd1b`**.

Warning

Before deleting the **`gfid-link`**, you must ensure that there are no hard links to the file present on that brick. If hard-links exist, you must delete them.

4. Trigger self-heal by running the following command:

```
# ls -l <file-path-on-gluster-mount>
```

or

```
# gluster volume heal VOLNAME
```

Appendix A. Revision History

Revision 3.4-3	**Mon Feb 04 2019**	**Red Hat Gluster Storage Documentation Team**

BZ#1484349 - Documented the procedure for adding new volumes in geo-replication
BZ#1486094 - Added warnings regarding Red Hat support for Red Hat Gluster Storage 3.4, 3.3, and 3.2 clients.
BZ#1536973 - Documented procedure to revoke the authorization of a client to access the Red Hat Gluster Storage trusted storage pool.
BZ#1564031 - Added information about snapshot metadata size.
BZ#1584459 - Documented the support for LVM Cache
BZ#1590496 - Documented the cluster.data-self-heal, cluster.entry-self-heal, and cluster.metadata-self-hea options in the Volume Options table.
BZ#1624397 - Removed EXT4 FS entries from examples in section Displaying Volume Status
BZ#1635120 - Documented the RHEL 7.5 support for Red Hat Gluster Storage 3.3.1.
BZ#1637490 - Documented the JBOD and RAID6 requirements for dedicated and chained arbiter configuration.
BZ#1639547 - Corrected error in replace host procedure.
BZ#1642378 - Various minor corrections in Configuration File section to improve usability.
BZ#1644480 - Added warnings about glusterd crash.
BZ#1648809 - Added RHEL 7 encrypted disk reference
BZ#1649331 - Added a new prerequisite to ensure that NFS is disabled on all volumes before running NFS Ganesha
BZ#1649798 - Revised network encryption chapter to improve usability.
BZ#1651541 - Documented the self heal prerequisite before converting existing replica 2 bricks to arbitrated
BZ#1659318 - Updated the Support Matrix table to include RHEL 7.6 information.
BZ#1659944 - Documented new default value for op-version.
BZ#1659965 - Documented the support for reader-thread-count option
BZ#1660456 - Removed mention of unsupported encryption translator.
BZ#1662889 - Removed the additional command to set slave volume in read-only mode to avoid being written by clients other than geo-replication.
BZ#1668278 - Documented the mandatory volume options user.cifs or user.smb to export gluster volume a SMB share via Samba.

Revision 3.4-2	**Wed Oct 31 2018**	**Red Hat Gluster Storage Documentation Team**

Content updates for Red Hat Gluster Storage 3.4 Batch 1 Update.

Revision 3.4-1	**Tue Sep 04 2018**	**Red Hat Gluster Storage Documentation Team**

Updates for Red Hat Gluster Storage 3.4 release.