Gluster Storage for Oracle® Linux

User's Guide for Release 3.12

F10040-04
June 2019

Oracle Legal Notices

Table of Contents

Preface

This document contains information about Gluster Storage for Oracle Linux Release 3.12. It describes the differences from the upstream version, includes notes on installing and configuring Gluster Storage for Oracle Linux, and provides a statement of what is supported.

Document generated on: 2019-06-06 (revision: 7598)

Audience

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

Related Documents

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

https://www.oracle.com/technetwork/server-storage/linux/documentation/index.html

Conventions

The following text conventions are used in this document:

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

Documentation Accessibility

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

Access to Oracle Support

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

Chapter 1 Introduction to Gluster Storage for Oracle Linux

Gluster is a scalable, distributed file system that aggregates disk storage resources from multiple servers into a single global namespace. This chapter provides introductory information about Gluster Storage for Oracle Linux Release 3.12.

1.1 About Gluster Storage for Oracle Linux

Gluster Storage for Oracle Linux Release 3.12 is based on the stable release of the upstream Gluster 3.12. Differences between Oracle versions of the software and upstream releases are limited to Oracle specific fixes and patches for specific bugs.

Note

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

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

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

Chapter 2 Installing Gluster Storage for Oracle Linux

This chapter discusses how to enable the repositories to install the Gluster Storage for Oracle Linux packages, and how to perform an installation of those packages.

2.1 Hardware and Network Requirements

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

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

Oracle provides support for Gluster Storage for Oracle Linux on 64-bit x86 hardware only.

A minimum node configuration is:

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

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

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

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

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

2.2 Operating System Requirements

Gluster Storage for Oracle Linux Release 3.12 is available for Oracle Linux 7 (x86_64) running the Red Hat Compatible Kernel (RHCK), the Unbreakable Enterprise Kernel Release 4 (UEK R4) or the Unbreakable Enterprise Kernel Release 5 (UEK R5). A minimum of Oracle Linux 7 Update 4 is required.

2.3 Enabling Access to the Gluster Storage for Oracle Linux Packages

The `glusterfs-server`, `glusterfs` and `glusterfs-fuse` packages are available on the Oracle Linux yum server in the `ol7_gluster312` repository, or on the Unbreakable Linux Network (ULN) in the `ol7_x86_64_gluster312` channel, however there are also dependencies across other repositories and channels, and these must also be enabled on each system where Gluster is installed.

If you are using the Oracle Linux yum server, you must enable the following repositories:

- `ol7_gluster312`
- `ol7_addons`
- `ol7_latest`
- `ol7_optional_latest`
- `ol7_UEKR5` or `ol7_UEKR4`

If you are using ULN, you must enable the following channels:

- `ol7_x86_64_gluster312`
- `ol7_x86_64_addons`
- `ol7_x86_64_latest`
- `ol7_x86_64_optional_latest`
- `ol7_x86_64_UEKR5` or `ol7_x86_64_UEKR4`

Enabling Repositories with ULN

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

To subscribe to the ULN channels:

1. Log in to https://linux.oracle.com with your ULN user name and password.
2. On the Systems tab, click the link named for the system in the list of registered machines.
3. On the System Details page, click **Manage Subscriptions**.
4. On the System Summary page, select each required channel from the list of available channels and click the right arrow to move the channel to the list of subscribed channels. Subscribe the system to the following channels:
   - `ol7_x86_64_gluster312`
   - `ol7_x86_64_addons`
   - `ol7_x86_64_latest`
   - `ol7_x86_64_optional_latest`
   - `ol7_x86_64_UEKR5` or `ol7_x86_64_UEKR4`
5. Click **Save Subscriptions**.

Enabling Repositories with the Oracle Linux Yum Server

To enable the required repositories on the Oracle Linux yum server, ensure that your system is up to date and that you have transitioned to use the modular yum repository configuration by installing the `oraclelinux-release-el7` package and running the `/usr/bin/ol_yum_configure.sh` script.

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

Install the `oracle-gluster-release-el7` release package to install appropriate yum repository configuration.

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

Enable the following repositories:

- `ol7_gluster312`
- `ol7_addons`
- `ol7_latest`
- `ol7_optional_latest`
- `ol7_UEKR5` or `ol7_UEKR4`

Use the `yum-config-manager` tool to update your yum configuration:

```
# yum-config-manager --enable ol7_gluster312 ol7_addons ol7_latest ol7_optional_latest ol7_UEKR5
```

You can now prepare the nodes for installation of the Gluster Storage for Oracle Linux packages. See Section 2.4, “Installing and Configuring Gluster”.

2.4 Installing and Configuring Gluster

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

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

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

2.4.1 Preparing Oracle Linux Nodes

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

To prepare Oracle Linux nodes:

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

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

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

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

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

   https://docs.oracle.com/cd/E52668_01/E54669/html/ol7-nettime.html

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

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

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

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

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

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

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

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

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

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

You can now install and configure the Gluster server.

2.4.2 Installing the Gluster Server

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

To install the Gluster server packages:

1. Install the `glusterfs-server` package.

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

2. Start and enable the Gluster server service:

```
# systemctl enable glusterd
# systemctl start glusterd
```

Chapter 3 Using Gluster Storage for Oracle Linux

This chapter discusses setting up Gluster trusted storage pools, Gluster volume types and setting them up, monitoring Gluster volumes, and accessing Gluster storage from an Oracle Linux or Microsoft Windows client system.

3.1 Creating the Gluster Trusted Storage Pool

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

To create a Gluster trusted storage pool:

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

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

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

```
# gluster peer status
```

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

```
# gluster pool list
```

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

```
# gluster peer detach hostname
```

3.2 Setting up Transport Layer Security (TLS)

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

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

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

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

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

section describes configuration for environments where certificates are signed by a CA and for when certificates are self-signed.

To configure TLS on nodes in a Gluster pool:

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

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

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

   To use self-signed certificates:

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

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

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

   To use CA-signed certificates:

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

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

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

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

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

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

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

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

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

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

```
# systemctl restart glusterd
```

3.3 Creating Gluster Volumes

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

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

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

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

In this example, *`brick_name`* is the file system that can be mounted from a client. For information on mounting a Gluster file system, see Section 3.6, “Accessing Gluster Volumes”.

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

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

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

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

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

Note that when a volume is configured, you can enable TLS on the volume to authenticate and encrypt connections between nodes that serve data for the volume, and for client systems that connect to the pool to access the volume. See Section 3.2, “Setting up Transport Layer Security (TLS)” for more information.

For more detailed information, see the Gluster upstream documentation.

3.3.1 Creating Distributed Volumes

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

Example 3.1 Creating a distributed volume

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

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

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

3.3.2 Creating Replicated Volumes

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

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

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

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

Note

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

Example 3.2 Creating a replicated volume

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

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

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

Example 3.3 Creating a replicated volume with an arbiter

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

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

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

3.3.3 Creating Distributed Replicated Volumes

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

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

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

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

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

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

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

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

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

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

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

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

```
# gluster volume create myvolume replica 3 \
  node{1,2,3}:/data/glusterfs/myvolume/mybrick/brick1 \
  node{1,2,3}:/data/glusterfs/myvolume/mybrick/brick2
```

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

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

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

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

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

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

3.3.4 Creating Dispersed Volumes

This section discusses creating a pool using dispersed volumes.

You set the volume redundancy level when you create a dispersed volume. The `redundancy` value sets how many bricks can be lost without interrupting the operation of the volume. The `redundancy` value

must be greater than `0`, and the total number of bricks must be greater than `2*redundancy`. A dispersed volume must have a minimum of three bricks.

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

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

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

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

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

3.3.5 Creating Distributed Dispersed Volumes

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

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

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

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

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

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

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

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

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

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

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

3.4 Managing Gluster Volumes

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

3.4.1 Setting Volume Options

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

`gluster volume set` *`volume_name option`*

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

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

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

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

3.4.2 Starting a Volume

To start a volume, use the the command:

`gluster volume start volume_name`

3.4.3 Stopping a Volume

To stop a volume, use the the command:

`gluster volume stop volume_name`

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

3.4.4 Self Healing a Replicated Volume

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

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

You can start a self-healing manually using:

```
# gluster volume heal myvolume
```

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

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

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

3.4.5 Expanding a Volume

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

To expand a volume:

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

```
$ gluster peer probe node4
```

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

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

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

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

   You can check the status of the volume rebalance using:

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

Example 3.11 Creating a distributed replicated volume and adding a node

This example creates a distributed replicated volume with three nodes and two bricks on each node. The volume is then extended to add a new node with an additional two bricks on the node. Note that when you add a new node to a replicated volume, you need to increase the `replica` count to the new number of nodes in the pool.

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

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

# gluster peer status
Number of Peers: 2

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

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

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

Hostname: node2
```

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

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

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

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

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

Example 3.12 Adding bricks to nodes in a distributed replicated volume

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

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

Volume Name: myvolume
Type: Distributed-Replicate
Volume ID: ...
Status: Started
Snapshot Count: 0
Number of Bricks: 4 x 4 = 16
Transport-type: tcp
Bricks:
```

```
Brick1: node1:/data/glusterfs/myvolume/mybrick/brick1
Brick2: node2:/data/glusterfs/myvolume/mybrick/brick1
Brick3: node3:/data/glusterfs/myvolume/mybrick/brick1
Brick4: node4:/data/glusterfs/myvolume/mybrick/brick1
Brick5: node1:/data/glusterfs/myvolume/mybrick/brick2
Brick6: node2:/data/glusterfs/myvolume/mybrick/brick2
Brick7: node3:/data/glusterfs/myvolume/mybrick/brick2
Brick8: node4:/data/glusterfs/myvolume/mybrick/brick2
Brick9: node1:/data/glusterfs/myvolume/mybrick/brick3
Brick10: node2:/data/glusterfs/myvolume/mybrick/brick3
Brick11: node3:/data/glusterfs/myvolume/mybrick/brick3
Brick12: node4:/data/glusterfs/myvolume/mybrick/brick3
Brick13: node1:/data/glusterfs/myvolume/mybrick/brick4
Brick14: node2:/data/glusterfs/myvolume/mybrick/brick4
Brick15: node3:/data/glusterfs/myvolume/mybrick/brick4
Brick16: node4:/data/glusterfs/myvolume/mybrick/brick4
Options Reconfigured:
transport.address-family: inet
nfs.disable: on
performance.client-io-threads: off.
# gluster volume rebalance myvolume start
volume rebalance: myvolume: success: Rebalance on myvolume has been started successfully.
Use rebalance status command to check status of the rebalance process.
ID: ...
# gluster volume rebalance myvolume status
...
volume rebalance: myvolume: success
```

3.4.6 Shrinking a Volume

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

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

To shrink a volume:

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

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

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

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

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

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

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

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

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

Example 3.13 Removing a node from a distributed replicated volume

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

```
# gluster volume info

Volume Name: myvolume
Type: Distributed-Replicate
Volume ID: ...
Status: Started
Snapshot Count: 0
Number of Bricks: 4 x 4 = 16
Transport-type: tcp
Bricks:
Brick1: node1:/data/glusterfs/myvolume/mybrick/brick1
Brick2: node2:/data/glusterfs/myvolume/mybrick/brick1
Brick3: node3:/data/glusterfs/myvolume/mybrick/brick1
Brick4: node4:/data/glusterfs/myvolume/mybrick/brick1
Brick5: node1:/data/glusterfs/myvolume/mybrick/brick2
Brick6: node2:/data/glusterfs/myvolume/mybrick/brick2
Brick7: node3:/data/glusterfs/myvolume/mybrick/brick2
Brick8: node4:/data/glusterfs/myvolume/mybrick/brick2
Brick9: node1:/data/glusterfs/myvolume/mybrick/brick3
Brick10: node2:/data/glusterfs/myvolume/mybrick/brick3
Brick11: node3:/data/glusterfs/myvolume/mybrick/brick3
Brick12: node4:/data/glusterfs/myvolume/mybrick/brick3
Brick13: node1:/data/glusterfs/myvolume/mybrick/brick4
Brick14: node2:/data/glusterfs/myvolume/mybrick/brick4
Brick15: node3:/data/glusterfs/myvolume/mybrick/brick4
Brick16: node4:/data/glusterfs/myvolume/mybrick/brick4
Options Reconfigured:
transport.address-family: inet
nfs.disable: on
performance.client-io-threads: off
# gluster volume remove-brick myvolume replica 3 \
  node4:/data/glusterfs/myvolume/mybrick/brick1 \
  node4:/data/glusterfs/myvolume/mybrick/brick2 \
  node4:/data/glusterfs/myvolume/mybrick/brick3 \
  node4:/data/glusterfs/myvolume/mybrick/brick4 \
  force
# gluster volume info

Volume Name: myvolume
Type: Distributed-Replicate
Volume ID: ...
Status: Started
Snapshot Count: 0
Number of Bricks: 4 x 3 = 12
Transport-type: tcp
Bricks:
Brick1: node1:/data/glusterfs/myvolume/mybrick/brick1
Brick2: node2:/data/glusterfs/myvolume/mybrick/brick1
Brick3: node3:/data/glusterfs/myvolume/mybrick/brick1
Brick4: node1:/data/glusterfs/myvolume/mybrick/brick2
Brick5: node2:/data/glusterfs/myvolume/mybrick/brick2
Brick6: node3:/data/glusterfs/myvolume/mybrick/brick2
Brick7: node1:/data/glusterfs/myvolume/mybrick/brick3
Brick8: node2:/data/glusterfs/myvolume/mybrick/brick3
Brick9: node3:/data/glusterfs/myvolume/mybrick/brick3
Brick10: node1:/data/glusterfs/myvolume/mybrick/brick4
Brick11: node2:/data/glusterfs/myvolume/mybrick/brick4
```

```
Brick12: node3:/data/glusterfs/myvolume/mybrick/brick4
Options Reconfigured:
transport.address-family: inet
nfs.disable: on
performance.client-io-threads: off
# gluster peer detach node4
peer detach: success
# gluster peer status
Number of Peers: 2

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

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

Example 3.14 Removing bricks from a distributed replicated volume

This example removes two bricks from a distributed replicated volume.

```
# gluster volume info

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

# gluster volume remove-brick myvolume \
```

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

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

3.4.7 Deleting a Volume

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

`gluster volume delete` *`volume_name`*

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

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

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

3.5 Monitoring Gluster Volumes

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

These are the main commands you use for monitoring volumes:

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

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

This section contains information on using these monitoring commands.

3.5.1 Using the Volume Status Command

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

To use the command, use the syntax:

```
gluster volume status volume_name options
```

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

Some commands that might be useful are:

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

Some more detailed examples that include output follow.

Example 3.15 Showing status information about bricks in a volume

This example displays status information about bricks in a volume.

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

Task Status of Volume myvolume
------------------------------------------------------------------------------
```

```
There are no active volume tasks
```

Example 3.16 Showing detailed status information about bricks in a volume

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

```
# gluster volume status myvolume detail
Status of volume: myvolume
------------------------------------------------------------------------------
Brick                : Brick node1:/data/glusterfs/myvolume/mybrick/brick
TCP Port             : 49154
RDMA Port            : 0
Online               : Y
Pid                  : 13553
File System          : xfs
Device               : /dev/vdb
Mount Options        : rw,relatime,attr2,inode64,noquota
Inode Size           : N/A
Disk Space Free      : 98.9GB
Total Disk Space     : 100.0GB
Inode Count          : 104857600
Free Inodes          : 104857526
------------------------------------------------------------------------------
...
Brick                : Brick node6:/data/glusterfs/myvolume/mybrick/brick
TCP Port             : 49152
RDMA Port            : 0
Online               : Y
Pid                  : 8913
File System          : xfs
Device               : /dev/vdb
Mount Options        : rw,relatime,attr2,inode64,noquota
Inode Size           : N/A
Disk Space Free      : 99.9GB
Total Disk Space     : 100.0GB
Inode Count          : 104857600
Free Inodes          : 104857574
```

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

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

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

----------------------------------------------
...
Brick : node6:/data/glusterfs/myvolume/mybrick/brick
Mallinfo
--------
Arena    : 9232384
```

```
Ordblks  : 184
Smblks   : 43
Hblks    : 9
Hblkhd   : 16203776
Usmblks  : 0
Fsmblks  : 4128
Uordblks : 3547696
Fordblks : 5684688
Keepcost : 30848

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

3.5.2 Using the Volume Profile Command

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

Note

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

To use the command, use the syntax:

`gluster volume profile` *`volume_name options`*

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

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

Some commands that might be useful are:

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

A more detailed example of using volume profiling follows.

Example 3.18 Using profiling to monitor a volume

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

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

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

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

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

Interval 5 Stats:

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

3.5.3 Using the Volume Top Command

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

To use the command, use the syntax:

`gluster volume top` *`volume_name options`*

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

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

Some commands that might be useful are:

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

`gluster volume top` *`volume_name`* `write` — Lists the files with the highest write calls on each brick in the volume.

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

Some more detailed examples that include output follow.

Example 3.19 Showing performance for all bricks in a volume

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

```
# gluster volume top myvolume read-perf  bs 2014 count 1024
Brick: node1:/data/glusterfs/myvolume/mybrick/brick
Throughput 1776.34 MBps time 0.0012 secs
Brick: node2:/data/glusterfs/myvolume/mybrick/brick
Throughput 1694.61 MBps time 0.0012 secs
Brick: node6:/data/glusterfs/myvolume/mybrick/brick
Throughput 1640.68 MBps time 0.0013 secs
Brick: node5:/data/glusterfs/myvolume/mybrick/brick
Throughput 1809.07 MBps time 0.0011 secs
Brick: node4:/data/glusterfs/myvolume/mybrick/brick
Throughput 1438.17 MBps time 0.0014 secs
Brick: node3:/data/glusterfs/myvolume/mybrick/brick
Throughput 1464.73 MBps time 0.0014 secs
# gluster volume top myvolume write-perf bs 2014 count 1024
Brick: node1:/data/glusterfs/myvolume/mybrick/brick
Throughput 779.42 MBps time 0.0026 secs
Brick: node4:/data/glusterfs/myvolume/mybrick/brick
Throughput 759.61 MBps time 0.0027 secs
Brick: node5:/data/glusterfs/myvolume/mybrick/brick
Throughput 763.26 MBps time 0.0027 secs
Brick: node6:/data/glusterfs/myvolume/mybrick/brick
Throughput 736.02 MBps time 0.0028 secs
Brick: node2:/data/glusterfs/myvolume/mybrick/brick
Throughput 751.85 MBps time 0.0027 secs
Brick: node3:/data/glusterfs/myvolume/mybrick/brick
Throughput 713.61 MBps time 0.0029 secs
```

Example 3.20 Showing performance for a brick

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

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

3.6 Accessing Gluster Volumes

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

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

Warning

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

3.6.1 Accessing Volumes using the Gluster Native Client (FUSE)

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

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

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

2. Install the Gluster native client packages:

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

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

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

4. If you have configured TLS for a volume, you may need to perform additional steps before a client system is able to mount the Gluster volume. See Section 3.2, “Setting up Transport Layer Security (TLS)” for more information. The following steps are required to complete client configuration for TLS:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

3.6.2 Accessing Volumes using Samba

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

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

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

https://docs.oracle.com/cd/E52668_01/E54669/html/ol7-about-samba.html

Setting up the Volume for Samba Access

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

To set up the volume for Samba access:

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

1. Install the Samba packages for Gluster:

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

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

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

3. Enable and start the Samba service:

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

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

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

   Restart the Samba service:

```
# systemctl restart smb
```

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

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

       passdb backend = tdbsam

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

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

   Restart the Samba service:

```
# systemctl restart smb
```

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

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

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

```
[gluster-myvolume]
comment = For samba share of volume myvolume
vfs objects = glusterfs
glusterfs:volume = myvolume
glusterfs:logfile = /var/log/samba/glusterfs-myvolume.%M.log
```

```
glusterfs:loglevel = 7
path = /
read only = no
guest ok = yes
kernel share modes = no
```

Note

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

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

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

To re-enable the hooks, you can run:

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

8. Disable enforcing mode on SELinux for any node where you have set up Samba access:

```
# setenforce 0
```

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

Testing SMB Access to a Volume

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

To test access to the volume using SMB:

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

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

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

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

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

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

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

Testing CIFS Access to a Volume

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

To test access to the volume using CIFS:

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

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

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

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

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

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

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

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

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

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

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

```
# chmod 600 /credfile
```

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

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

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

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

Accessing the Volume from Microsoft Windows

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

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

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

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

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

Gluster Terminology

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

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

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

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

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

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

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

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

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