Gluster File System 3.3.0 Administration Guide

Using Gluster File System

GlusterFS Developers

Gluster File System 3.3.0 Administration Guide
Using Gluster File System
Edition 1

Author GlusterFS Developers *gluster-devel@nongnu.org*

This guide describes Gluster File System (GlusterFS) and provides information on how to configure, operate, and manage GlusterFS.

Preface

This guide describes how to configure, operate, and manage Gluster File System (GlusterFS).

1. Audience

This guide is intended for Systems Administrators interested in configuring and managing GlusterFS.

This guide assumes that you are familiar with the Linux operating system, concepts of File System, GlusterFS concepts, and GlusterFS Installation

2. License

The License information is available at *http://www.redhat.com/licenses/rhel_rha_eula.html*.

3. Document Conventions

This manual uses several conventions to highlight certain words and phrases and draw attention to specific pieces of information.

In PDF and paper editions, this manual uses typefaces drawn from the *Liberation Fonts*[1] set. The Liberation Fonts set is also used in HTML editions if the set is installed on your system. If not, alternative but equivalent typefaces are displayed. Note: Red Hat Enterprise Linux 5 and later includes the Liberation Fonts set by default.

3.1. Typographic Conventions

Four typographic conventions are used to call attention to specific words and phrases. These conventions, and the circumstances they apply to, are as follows.

`Mono-spaced Bold`

Used to highlight system input, including shell commands, file names and paths. Also used to highlight keycaps and key combinations. For example:

> To see the contents of the file **`my_next_bestselling_novel`** in your current working directory, enter the **`cat my_next_bestselling_novel`** command at the shell prompt and press **`Enter`** to execute the command.

The above includes a file name, a shell command and a keycap, all presented in mono-spaced bold and all distinguishable thanks to context.

Key combinations can be distinguished from keycaps by the hyphen connecting each part of a key combination. For example:

> Press **`Enter`** to execute the command.
>
> Press **`Ctrl`**+**`Alt`**+**`F2`** to switch to the first virtual terminal. Press **`Ctrl`**+**`Alt`**+**`F1`** to return to your X-Windows session.

The first paragraph highlights the particular keycap to press. The second highlights two key combinations (each a set of three keycaps with each set pressed simultaneously).

[1] https://fedorahosted.org/liberation-fonts/

If source code is discussed, class names, methods, functions, variable names and returned values mentioned within a paragraph will be presented as above, in **mono-spaced bold**. For example:

> File-related classes include **filesystem** for file systems, **file** for files, and **dir** for directories. Each class has its own associated set of permissions.

Proportional Bold

This denotes words or phrases encountered on a system, including application names; dialog box text; labeled buttons; check-box and radio button labels; menu titles and sub-menu titles. For example:

> Choose **System** → **Preferences** → **Mouse** from the main menu bar to launch **Mouse Preferences**. In the **Buttons** tab, click the **Left-handed mouse** check box and click **Close** to switch the primary mouse button from the left to the right (making the mouse suitable for use in the left hand).
>
> To insert a special character into a **gedit** file, choose **Applications** → **Accessories** → **Character Map** from the main menu bar. Next, choose **Search** → **Find…** from the **Character Map** menu bar, type the name of the character in the **Search** field and click **Next**. The character you sought will be highlighted in the **Character Table**. Double-click this highlighted character to place it in the **Text to copy** field and then click the **Copy** button. Now switch back to your document and choose **Edit** → **Paste** from the **gedit** menu bar.

The above text includes application names; system-wide menu names and items; application-specific menu names; and buttons and text found within a GUI interface, all presented in proportional bold and all distinguishable by context.

Mono-spaced Bold Italic or ***Proportional Bold Italic***

Whether mono-spaced bold or proportional bold, the addition of italics indicates replaceable or variable text. Italics denotes text you do not input literally or displayed text that changes depending on circumstance. For example:

> To connect to a remote machine using ssh, type **ssh *username*@*domain.name*** at a shell prompt. If the remote machine is **example.com** and your username on that machine is john, type **ssh john@example.com**.
>
> The **mount -o remount *file-system*** command remounts the named file system. For example, to remount the **/home** file system, the command is **mount -o remount /home**.
>
> To see the version of a currently installed package, use the **rpm -q *package*** command. It will return a result as follows: ***package-version-release***.

Note the words in bold italics above — username, domain.name, file-system, package, version and release. Each word is a placeholder, either for text you enter when issuing a command or for text displayed by the system.

Aside from standard usage for presenting the title of a work, italics denotes the first use of a new and important term. For example:

> Publican is a *DocBook* publishing system.

3.2. Pull-quote Conventions

Terminal output and source code listings are set off visually from the surrounding text.

Output sent to a terminal is set in **mono-spaced roman** and presented thus:

```
books        Desktop   documentation  drafts  mss    photos   stuff  svn
books_tests  Desktop1  downloads      images  notes  scripts  svgs
```

Source-code listings are also set in **mono-spaced roman** but add syntax highlighting as follows:

```
package org.jboss.book.jca.ex1;

import javax.naming.InitialContext;

public class ExClient
{
   public static void main(String args[])
       throws Exception
   {
      InitialContext iniCtx = new InitialContext();
      Object         ref    = iniCtx.lookup("EchoBean");
      EchoHome       home   = (EchoHome) ref;
      Echo           echo   = home.create();

      System.out.println("Created Echo");

      System.out.println("Echo.echo('Hello') = " + echo.echo("Hello"));
   }
}
```

3.3. Notes and Warnings

Finally, we use three visual styles to draw attention to information that might otherwise be overlooked.

Note

Notes are tips, shortcuts or alternative approaches to the task at hand. Ignoring a note should have no negative consequences, but you might miss out on a trick that makes your life easier.

Important

Important boxes detail things that are easily missed: configuration changes that only apply to the current session, or services that need restarting before an update will apply. Ignoring a box labeled 'Important' will not cause data loss but may cause irritation and frustration.

Warning

Warnings should not be ignored. Ignoring warnings will most likely cause data loss.

4. We Need Feedback!

If you find any issues, please open a bug on our Bugzilla - *https://bugzilla.redhat.com/enter_bug.cgi?product=GlusterFS*

http://www.gluster.org/interact/mailinglists/ - For details about mailing lists check out our community page

If you want live help, join us on #gluster on freenode (IRC channel)

Introducing Gluster File System

GlusterFS is an open source, clustered file system capable of scaling to several petabytes and handling thousands of clients. GlusterFS can be flexibly combined with commodity physical, virtual, and cloud resources to deliver highly available and performant enterprise storage at a fraction of the cost of traditional solutions.

GlusterFS clusters together storage building blocks over Infiniband RDMA and/or TCP/IP interconnect, aggregating disk and memory resources and managing data in a single global namespace. GlusterFS is based on a stackable user space design, delivering exceptional performance for diverse workloads.

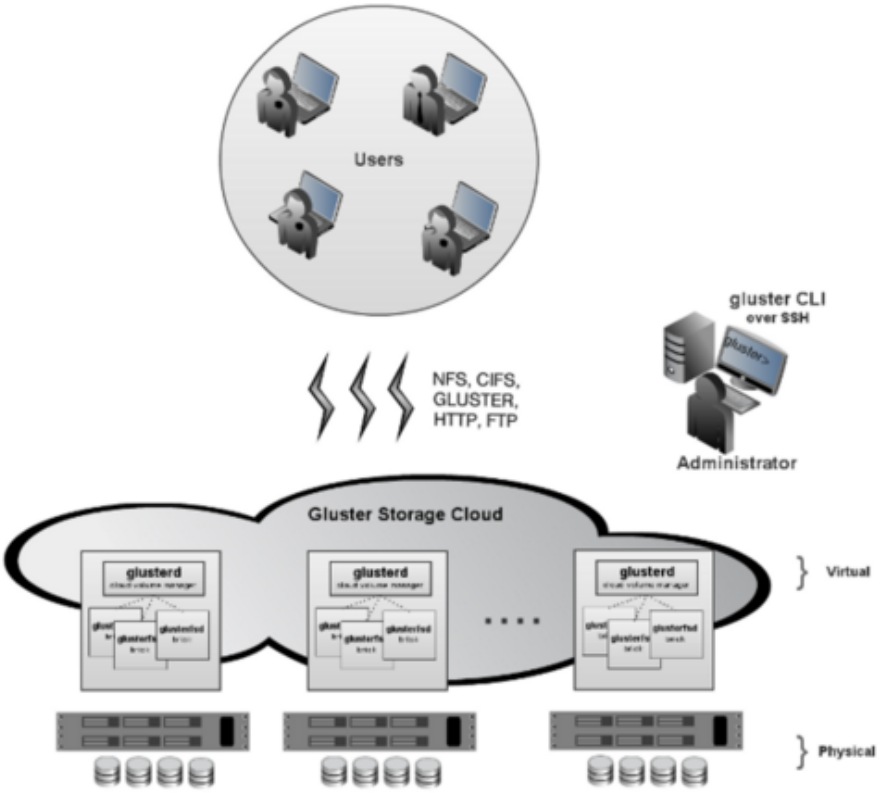


Figure 1.1. Virtualized Cloud Environments

GlusterFS is designed for today's high-performance, virtualized cloud environments. Unlike traditional data centers, cloud environments require multi-tenancy along with the ability to grow or shrink resources on demand. Enterprises can scale capacity, performance, and availability on demand, with no vendor lock-in, across on-premise, public cloud, and hybrid environments.

GlusterFS is in production at thousands of enterprises spanning media, healthcare, government, education, web 2.0, and financial services. The following table lists the commercial offerings and its documentation location:

Product	Documentation Location
Red Hat Storage Software Appliance	*http://docs.redhat.com/docs/en-US/Red_Hat_Storage_Software_Appliance/index.html*
Red Hat Virtual	*http://docs.redhat.com/docs/en-US/Red_Hat_Virtual_Storage_Appliance/index.html*

Product	Documentation Location
Storage Appliance	
Red Hat Storage	*http://docs.redhat.com/docs/en-US/Red_Hat_Storage/index.html*

Managing the glusterd Service

After installing GlusterFS, you must start glusterd service. The glusterd service serves as the Gluster elastic volume manager, overseeing glusterfs processes, and co-ordinating dynamic volume operations, such as adding and removing volumes across multiple storage servers non-disruptively.

This section describes how to start the glusterd service in the following ways:

- *Section 2.1, “Starting and Stopping glusterd Manually”*
- *Section 2.2, “Starting glusterd Automatically”*

Note

You must start glusterd on all GlusterFS servers.

2.1. Starting and Stopping glusterd Manually

This section describes how to start and stop glusterd manually

- To start glusterd manually, enter the following command:

  ```
  # /etc/init.d/glusterd start
  ```

- To stop glusterd manually, enter the following command:

  ```
  # /etc/init.d/glusterd stop
  ```

2.2. Starting glusterd Automatically

This section describes how to configure the system to automatically start the glusterd service every time the system boots.

2.2.1. Red Hat-based Systems

To configure Red Hat-based systems to automatically start the glusterd service every time the system boots, enter the following from the command line:

```
# chkconfig glusterd on
```

2.2.2. Debian-based Systems

To configure Debian-based systems to automatically start the glusterd service every time the system boots, enter the following from the command line:

```
# update-rc.d glusterd defaults
```

2.2.3. Systems Other than Red Hat and Debain

To configure systems other than Red Hat or Debian to automatically start the glusterd service every time the system boots, enter the following entry to the */etc/rc.local* file:

```
# echo "glusterd" >> /etc/rc.local
```

Using the Gluster Console Manager – Command Line Utility

The Gluster Console Manager is a single command line utility that simplifies configuration and management of your storage environment. The Gluster Console Manager is similar to the LVM (Logical Volume Manager) CLI or ZFS Command Line Interface, but it works in sync with multiple storage servers. You can use the Gluster Console Manager while volumes are mounted and active too. Gluster automatically synchronizes volume configuration information across all Gluster servers.

Using the Gluster Console Manager, you can create new volumes, start volumes, and stop volumes, as required. You can also add bricks to volumes, remove bricks from existing volumes, as well as change volume settings (such as some translator specific options), among other operations.

You can also use these CLI commands to create scripts for automation, as well as use the commands as an API to allow integration with third-party applications.

Running the Gluster Console Manager

You can run the Gluster Console Manager on any GlusterFS server either by invoking the commands or by running the Gluster CLI in interactive mode. You can also use the gluster command remotely using SSH.

- To run commands directly:

  # gluster peer *command*

  For example:

  # gluster peer status

- To run the Gluster Console Manager in interactive mode

  # gluster

  You can execute gluster commands from the Console Manager prompt:

  gluster> *command*

  For example, to view the status of the peer server:

  # **gluster**

  gluster > peer status

  Display the status of the peer.

With any 'gluster' installation, to check all the supported CLI commands, use **'gluster help'** .

Setting up Trusted Storage Pools

Before you can configure a GlusterFS volume, you must create a trusted storage pool consisting of the storage servers that provides bricks to a volume.

A storage pool is a trusted network of storage servers. When you start the first server, the storage pool consists of that server alone. To add additional storage servers to the storage pool, you can use the probe command from a storage server that is already part of the trusted storage pool.

Note

Do not self-probe the first server/localhost.

The glusterd service must be running on all storage servers that you want to add to the storage pool. See *Chapter 2, Managing the glusterd Service* for more information.

4.1. Adding Servers to Trusted Storage Pool

To create a trusted storage pool, add servers to the trusted storage pool

1. The hostnames used to create the storage pool must be resolvable by DNS. Also make sure that firewall is not blocking the probe requests/replies. (iptables -F)

   To add a server to the storage pool:

   # gluster peer probe *server*

   For example, to create a trusted storage pool of four servers, add three servers to the storage pool from server1:

   ```
   # gluster peer probe server2
   Probe successful

   # gluster peer probe server3
   Probe successful

   # gluster peer probe server4
   Probe successful
   ```

2. Verify the peer status from the first server using the following commands:

   ```
   # 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)
```

4.2. Removing Servers from the Trusted Storage Pool

To remove a server from the storage pool:

 # gluster peer detach *server*

For example, to remove server4 from the trusted storage pool:

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

Setting up GlusterFS Server Volumes

A 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 gluster management operations are performed on the volume.

To create a new volume in your storage environment, specify the bricks that comprise the volume. After you have created a new volume, you must start it before attempting to mount it.

- Volumes of the following types can be created in your storage environment:
  - Distributed - Distributed volumes distributes files throughout the bricks in the volume. You can use distributed volumes where the requirement is to scale storage and the redundancy is either not important or is provided by other hardware/software layers. For more information, see *Section 5.1, "Creating Distributed Volumes"* .
  - Replicated – Replicated volumes replicates files across bricks in the volume. You can use replicated volumes in environments where high-availability and high-reliability are critical. For more information, see *Section 5.2, "Creating Replicated Volumes "*.
  - Striped – Striped volumes stripes data across bricks in the volume. For best results, you should use striped volumes only in high concurrency environments accessing very large files. For more information, see *Section 5.3, "Creating Striped Volumes"*.
  - Distributed Striped - Distributed striped volumes stripe data across two or more nodes in the cluster. You should use distributed striped volumes where the requirement is to scale storage and in high concurrency environments accessing very large files is critical. For more information, see *Section 5.4, "Creating Distributed Striped Volumes "*.
  - Distributed Replicated - Distributed replicated volumes distributes files across replicated bricks in the volume. You can use distributed replicated volumes in environments where the requirement is to scale storage and high-reliability is critical. Distributed replicated volumes also offer improved read performance in most environments. For more information, see *Section 5.5, "Creating Distributed Replicated Volumes "*.
  - Distributed Striped Replicated – Distributed striped replicated volumes distributes striped data across replicated bricks in the cluster. For best results, you should use distributed striped replicated volumes in highly concurrent environments where parallel access of very large files and performance is critical. In this release, configuration of this volume type is supported only for Map Reduce workloads. For more information, see *Section 5.6, "Creating Distributed Striped Replicated Volumes "*.
  - Striped Replicated – Striped replicated volumes stripes data across replicated bricks in the cluster. For best results, you should use striped replicated volumes in highly concurrent environments where there is parallel access of very large files and performance is critical. In this release, configuration of this volume type is supported only for Map Reduce workloads. For more information, see *Section 5.7, "Creating Striped Replicated Volumes "*.

To create a new volume

- Create a new volume :

  ```
  # gluster volume create NEW-VOLNAME [stripe COUNT | replica COUNT]
  [transport [tcp | rdma | tcp,rdma]] NEW-BRICK1 NEW-BRICK2 NEW-BRICK3...
  ```

  For example, to create a volume called test-volume consisting of server3:/exp3 and server4:/exp4:

```
# gluster volume create test-volume server3:/exp3 server4:/exp4
Creation of test-volume has been successful
Please start the volume to access data.
```

5.1. Creating Distributed Volumes

In a distributed volumes files are spread randomly across the bricks in the volume. Use distributed volumes where you need to scale storage and redundancy is either not important or is provided by other hardware/software layers.

Note

Disk/server failure in distributed volumes can result in a serious loss of data because directory contents are spread randomly across the bricks in the volume.

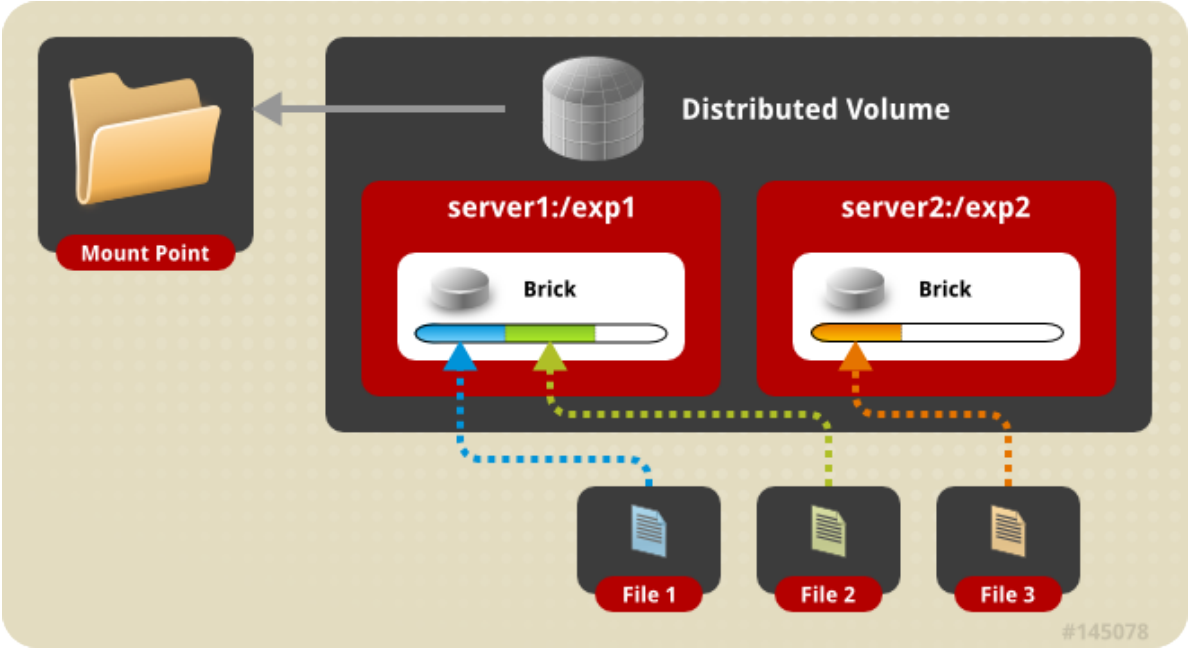


Figure 5.1. Illustration of a Distributed Volume

To create a distributed volume

1. Create a trusted storage pool as described earlier in *Section 4.1, "Adding Servers to Trusted Storage Pool"*.

2. Create the distributed volume:

   # gluster volume create *NEW-VOLNAME* [transport [tcp | rdma | tcp,rdma]] *NEW-BRICK...*

   For example, to create a distributed volume with four storage servers using tcp:

   ```
   # gluster volume create test-volume server1:/exp1 server2:/exp2 server3:/exp3 server4:/
   exp4
   ```

```
Creation of test-volume has been successful
Please start the volume to access data.
```

(Optional) You can display the volume information:

```
# gluster volume info
Volume Name: test-volume
Type: Distribute
Status: Created
Number of Bricks: 4
Transport-type: tcp
Bricks:
Brick1: server1:/exp1
Brick2: server2:/exp2
Brick3: server3:/exp3
Brick4: server4:/exp4
```

For example, to create a distributed volume with four storage servers over InfiniBand:

```
# gluster volume create test-volume transport rdma server1:/exp1 server2:/exp2 server3:/
exp3 server4:/exp4
Creation of test-volume has been successful
Please start the volume to access data.
```

If the transport type is not specified, *tcp* is used as the default. You can also set additional options if required, such as auth.allow or auth.reject. For more information, see *Section 7.1, “Tuning Volume Options”*

Note

Make sure you start your volumes before you try to mount them or else client operations after the mount will hang, see *Section 5.8, “Starting Volumes ”* for details.

5.2. Creating Replicated Volumes

Replicated volumes create copies of files across multiple bricks in the volume. You can use replicated volumes in environments where high-availability and high-reliability are critical.

Note

The number of bricks should be equal to of 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.

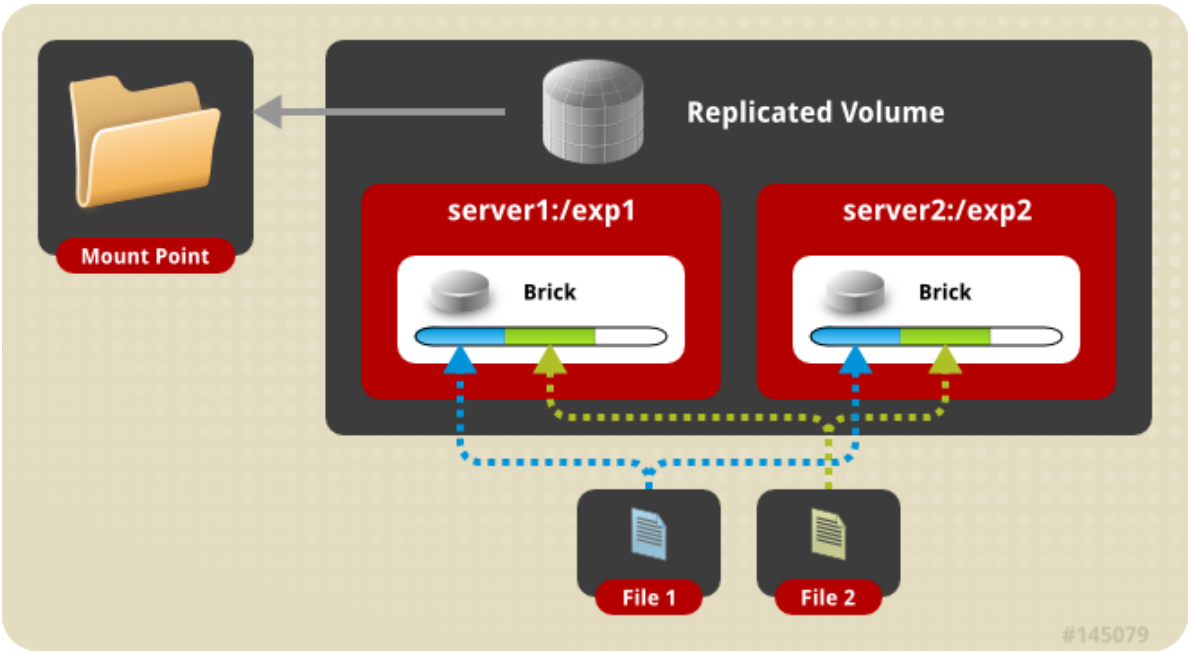


Figure 5.2. Illustration of a Replicated Volume

To create a replicated volume

1. Create a trusted storage pool as described earlier in *Section 4.1, "Adding Servers to Trusted Storage Pool"*.
2. Create the replicated volume:

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

   For example, to create a replicated volume with two storage servers:

   ```
   # gluster volume create test-volume replica 2 transport tcp server1:/exp1 server2:/exp2
   Creation of test-volume has been successful
   Please start the volume to access data.
   ```

   If the transport type is not specified, *tcp* is used as the default. You can also set additional options if required, such as auth.allow or auth.reject. For more information, see *Section 7.1, "Tuning Volume Options"*

   Note

   Make sure you start your volumes before you try to mount them or else client operations after the mount will hang, see *Section 5.8, "Starting Volumes "* for details.

5.3. Creating Striped Volumes

Striped volumes stripes data across bricks in the volume. For best results, you should use striped volumes only in high concurrency environments accessing very large files.

Note

The number of bricks should be a equal to the stripe count for a striped volume.

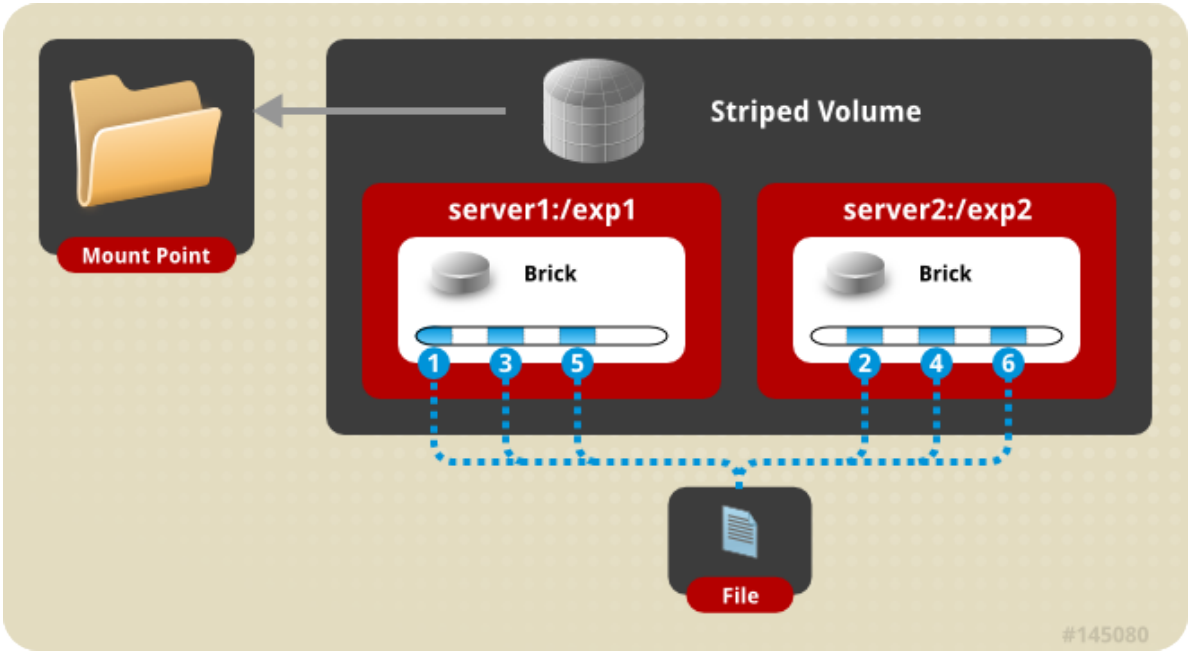


Figure 5.3. Illustration of a Striped Volume

To create a striped volume

1. Create a trusted storage pool as described earlier in *Section 4.1, "Adding Servers to Trusted Storage Pool"*.

2. Create the striped volume:

   `# gluster volume create NEW-VOLNAME [stripe COUNT] [transport [tcp | rdma | tcp,rdma]] NEW-BRICK...`

   For example, to create a striped volume across two storage servers:

   ```
   # gluster volume create test-volume stripe 2 transport tcp server1:/exp1 server2:/exp2
   Creation of test-volume has been successful
   Please start the volume to access data.
   ```

   If the transport type is not specified, *tcp* is used as the default. You can also set additional options if required, such as auth.allow or auth.reject. For more information, see *Section 7.1, "Tuning Volume Options"*

Note

Make sure you start your volumes before you try to mount them or else client operations after the mount will hang, see *Section 5.8, "Starting Volumes "* for details.

5.4. Creating Distributed Striped Volumes

Distributed striped volumes stripes files across two or more nodes in the cluster. For best results, you should use distributed striped volumes where the requirement is to scale storage and in high concurrency environments accessing very large files is critical.

Note

The number of bricks should be a multiple of the stripe count for a distributed striped volume.

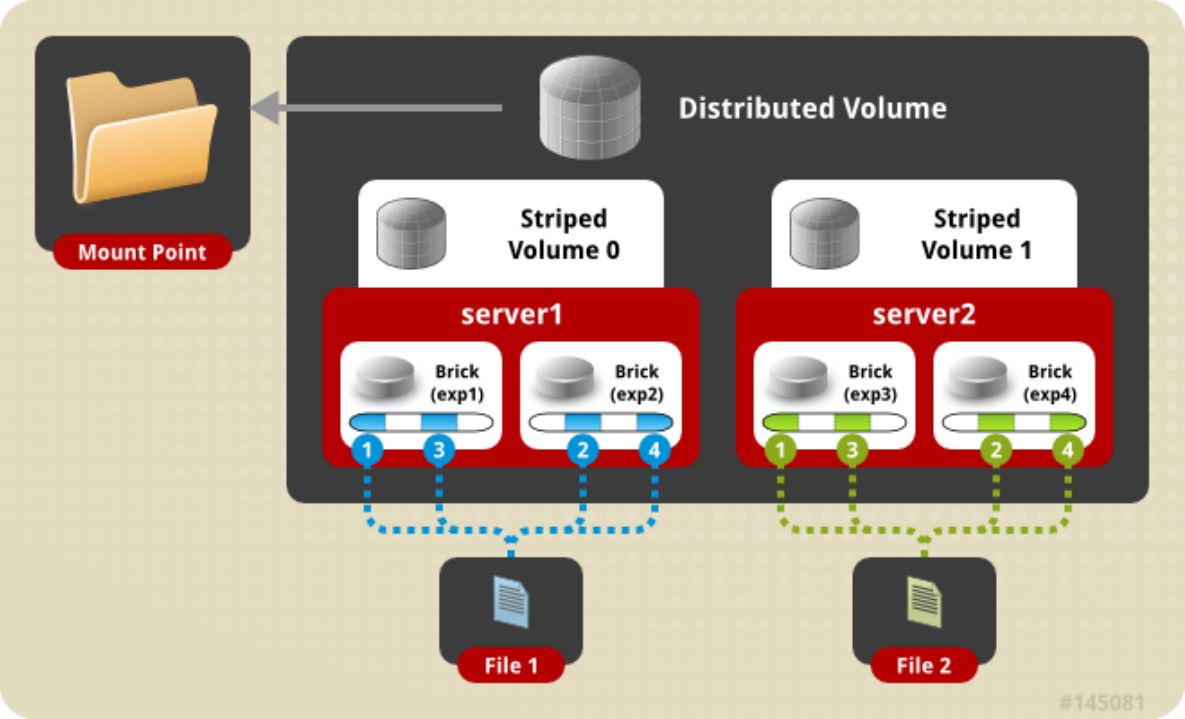


Figure 5.4. Illustration of a Distributed Striped Volume

To create a distributed striped volume

1. Create a trusted storage pool as described earlier in *Section 4.1, "Adding Servers to Trusted Storage Pool"*.

2. Create the distributed striped volume:

```
# gluster volume create NEW-VOLNAME [stripe COUNT] [transport [tcp |
rdma | tcp,rdma]] NEW-BRICK...
```

For example, to create a distributed striped volume across eight storage servers:

```
# gluster volume create test-volume stripe 4 transport tcp server1:/exp1 server2:/exp2
 server3:/exp3 server4:/exp4 server5:/exp5 server6:/exp6 server7:/exp7 server8:/exp8
Creation of test-volume has been successful
Please start the volume to access data.
```

If the transport type is not specified, *tcp* is used as the default. You can also set additional options if required, such as auth.allow or auth.reject. For more information, see *Section 7.1, “Tuning Volume Options”*

Note

Make sure you start your volumes before you try to mount them or else client operations after the mount will hang, see *Section 5.8, “Starting Volumes ”* for details.

5.5. Creating Distributed Replicated Volumes

Distributes files across replicated bricks in the volume. You can use distributed replicated volumes in environments where the requirement is to scale storage and high-reliability is critical. Distributed replicated volumes also offer improved read performance in most environments.

Note

The number of bricks should 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 volume-wide distribute set. To make sure 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.

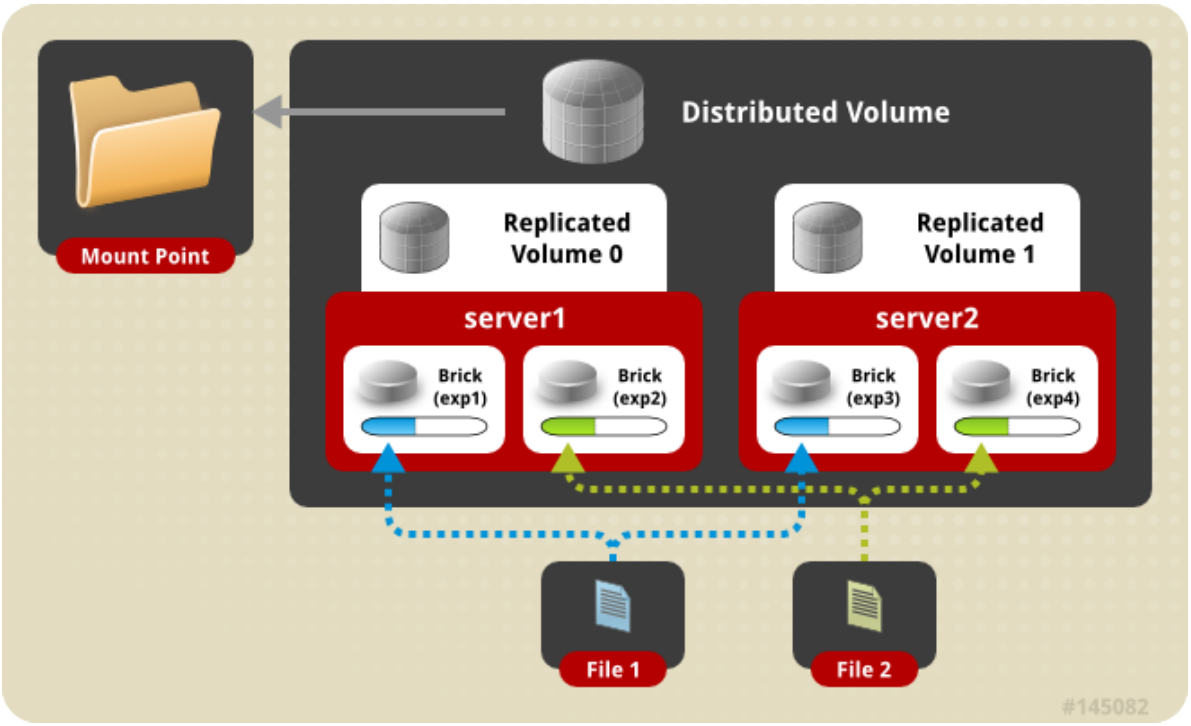


Figure 5.5. Illustration of a Distributed Replicated Volume

To create a distributed replicated volume

1. Create a trusted storage pool as described earlier in *Section 4.1, "Adding Servers to Trusted Storage Pool"*.

2. Create the distributed replicated volume:

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

   For example, four node distributed (replicated) volume with a two-way mirror:

   ```
   # gluster volume create test-volume replica 2 transport tcp server1:/exp1 server2:/exp2
    server3:/exp3 server4:/exp4
   Creation of test-volume has been successful
   Please start the volume to access data.
   ```

   For example, to create a six node distributed (replicated) volume with a two-way mirror:

   ```
   # gluster volume create test-volume replica 2 transport tcp server1:/exp1 server2:/exp2
    server3:/exp3 server4:/exp4 server5:/exp5 server6:/exp6
   Creation of test-volume has been successful
   Please start the volume to access data.
   ```

   If the transport type is not specified, *tcp* is used as the default. You can also set additional options if required, such as auth.allow or auth.reject. For more information, see *Section 7.1, "Tuning Volume Options"*

Note

Make sure you start your volumes before you try to mount them or else client operations after the mount will hang, see *Section 5.8, "Starting Volumes "* for details.

5.6. Creating Distributed Striped Replicated Volumes

Distributed striped replicated volumes distributes striped data across replicated bricks in the cluster. For best results, you should use distributed striped replicated volumes in highly concurrent environments where parallel access of very large files and performance is critical. In this release, configuration of this volume type is supported only for Map Reduce workloads.

Note

The number of bricks should be a multiples of number of stripe count and replica count for a distributed striped replicated volume.

To create a distributed striped replicated volume

1. Create a trusted storage pool as described earlier in *Section 4.1, "Adding Servers to Trusted Storage Pool"*.

2. Create a distributed striped replicated volume using the following command:

   `# gluster volume create NEW-VOLNAME [stripe COUNT] [replica COUNT] [transport [tcp | rdma | tcp,rdma]] NEW-BRICK...`

   For example, to create a distributed replicated striped volume across eight storage servers:

   ```
   # gluster volume create test-volume stripe 2 replica 2 transport tcp server1:/exp1
    server2:/exp2 server3:/exp3 server4:/exp4 server5:/exp5 server6:/exp6 server7:/exp7
    server8:/exp8
   Creation of test-volume has been successful
   Please start the volume to access data.
   ```

   If the transport type is not specified, *tcp* is used as the default. You can also set additional options if required, such as auth.allow or auth.reject. For more information, see *Section 7.1, "Tuning Volume Options"*

Note

Make sure you start your volumes before you try to mount them or else client operations after the mount will hang, see *Section 5.8, "Starting Volumes "* for details.

5.7. Creating Striped Replicated Volumes

Striped replicated volumes stripes data across replicated bricks in the cluster. For best results, you should use striped replicated volumes in highly concurrent environments where there is parallel access of very large files and performance is critical. In this release, configuration of this volume type is supported only for Map Reduce workloads.

Note

The number of bricks should be a multiple of the replicate count and stripe count for a striped replicated volume.

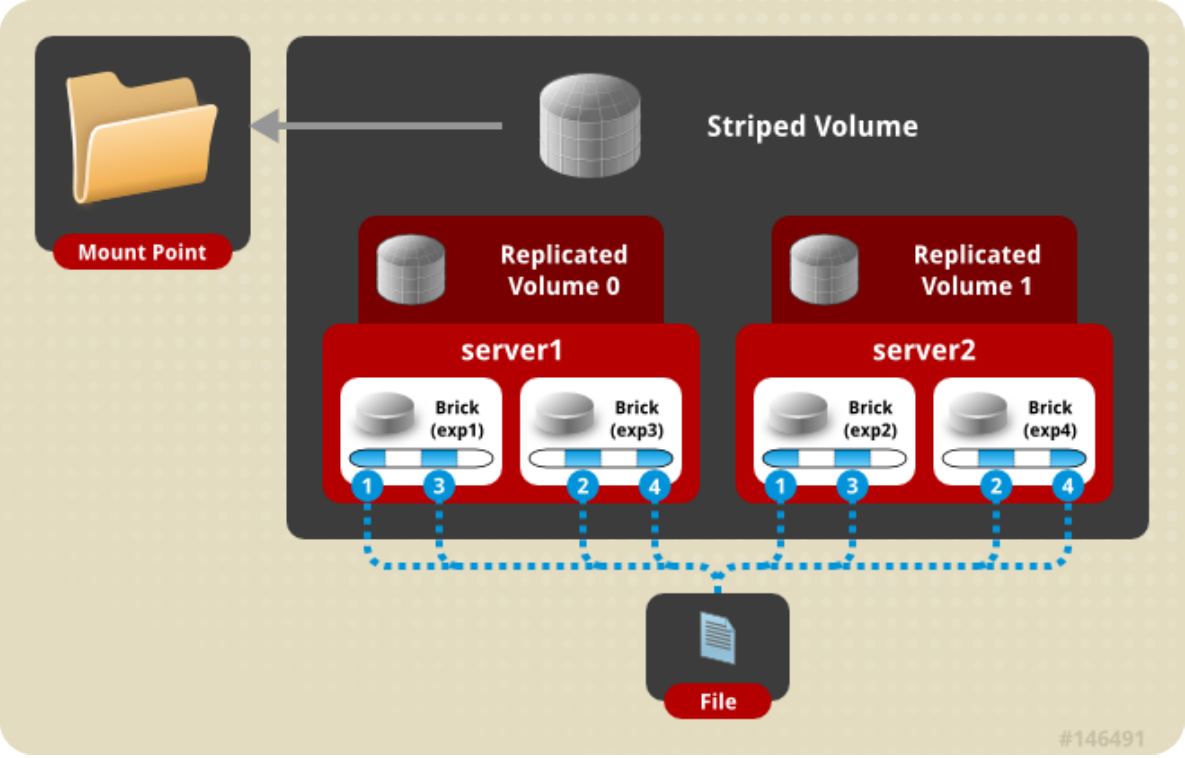


Figure 5.6. Illustration of a Striped Replicated Volume

To create a striped replicated volume

1. Create a trusted storage pool consisting of the storage servers that will comprise the volume.

   For more information, see *Section 4.1, "Adding Servers to Trusted Storage Pool"*.

2. Create a striped replicated volume :

   # gluster volume create *NEW-VOLNAME* [stripe *COUNT*] [replica *COUNT*] [transport [tcp | rdma | tcp,rdma]] *NEW-BRICK...*

   For example, to create a striped replicated volume across four storage servers:

```
# gluster volume create test-volume stripe 2 replica 2 transport tcp server1:/exp1
 server2:/exp2 server3:/exp3 server4:/exp4
Creation of test-volume has been successful
Please start the volume to access data.
```

   To create a striped replicated volume across six storage servers:

```
# gluster volume create test-volume stripe 3 replica 2 transport tcp server1:/exp1
 server2:/exp2 server3:/exp3 server4:/exp4 server5:/exp5 server6:/exp6
Creation of test-volume has been successful
Please start the volume to access data.
```

   If the transport type is not specified, *tcp* is used as the default. You can also set additional options if required, such as auth.allow or auth.reject. For more information, see *Section 7.1, "Tuning Volume Options"*

Note

Make sure you start your volumes before you try to mount them or else client operations after the mount will hang, see *Section 5.8, "Starting Volumes "* for details.

5.8. Starting Volumes

You must start your volumes before you try to mount them.

To start a volume

- Start a volume:

  # gluster volume start *VOLNAME*

  For example, to start test-volume:

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

Accessing Data - Setting Up GlusterFS Client

Gluster volumes can be accessed in multiple ways. One can use Gluster Native Client method for high concurrency, performance and transparent failover in GNU/Linux clients. Gluster exports volumes using NFS v3 protocol too.

CIFS can also be used to access volumes by exporting the Gluster Native mount point as a samba export.

6.1. Gluster Native Client

The Gluster Native Client is a FUSE-based client running in user space. Gluster Native Client is the recommended method for accessing volumes if all the clustered features of GlusterFS has to be utilized.

This section introduces the Gluster Native Client and explains how to install the software on client machines. This section also describes how to mount volumes on clients (both manually and automatically).

6.1.1. Installing the Gluster Native Client

Gluster Native Client has a dependancy on FUSE module. To make sure FUSE module is loaded, execute below commands:

1. Add the FUSE loadable kernel module (LKM) to the Linux kernel:

   ```
   # modprobe fuse
   ```

2. Verify that the FUSE module is loaded:

   ```
   # dmesg | grep -i fuse
   ```

   ```
   fuse init (API version 7.13)
   ```

6.1.1.1. Installing on RPM Based Distributions

To install Gluster Native Client on RPM distribution-based systems

1. Install required prerequisites on the client using the following command:

   ```
   $ sudo yum -y install fuse fuse-libs
   ```

2. Download the latest glusterfs, glusterfs-fuse RPM files on each client. The glusterfs package contains the GlusterFS Binary and required libraries. The glusterfs-fuse package contains the FUSE plugin (in GlusterFS terms, its called Translator) required for mounting.

   Note

   Install 'glusterfs-rdma' RPM if RDMA support is required. 'glusterfs-rdma' contains RDMA transport module for Infiniband interconnect.

You can download the software at *http://bits.gluster.com/gluster/glusterfs/3.3.0/x86_64/*.

3. Install Gluster Native Client on the client.

```
$ sudo rpm -i glusterfs-3.3.0-1.x86_64.rpm

$ sudo rpm -i glusterfs-fuse-3.3.0-1.x86_64.rpm

$ sudo rpm -i glusterfs-rdma-3.3.0-1.x86_64.rpm
```

6.1.1.2. Installing on Debian-based Distributions

To install Gluster Native Client on Debian-based distributions

1. Download the latest GlusterFS .deb file.

   You can download the software at *http://www.gluster.org/download/*.

2. Uninstall GlusterFS v3.1.x/v3.2.x (or an earlier version) from the client using the following command:

   ```
   $ sudo dpkg -r glusterfs
   ```

   (Optional) Run **`$ sudo dpkg -purge glusterfs`** to purge the configuration files.

3. Install Gluster Native Client on the client using the following command:

   ```
   $ sudo dpkg -i glusterfs-$version.deb
   ```

   For example:

   ```
   $ sudo dpkg -i glusterfs-3.3.0.deb
   ```

6.1.1.3. Performing a Source Installation

To build and install Gluster Native Client from the source code

1. Create a new directory using the following commands:

   ```
   # mkdir glusterfs

   # cd glusterfs
   ```

2. Download the source code.

   You can download the source at *http://www.gluster.org/download/*.

3. Extract the source code using the following command:

   ```
   # tar -xvzf glusterfs-3.3.0.tar.gz
   ```

4. Run the configuration utility using the following command:

   ```
   # ./configure

   ...

   GlusterFS configure summary

   ===========================
   ```

```
FUSE client : yes

Infiniband verbs : yes

epoll IO multiplex : yes

argp-standalone : no

fusermount : no

readline : yes
```

Note

The configuration summary shown above is sample, it can vary depending on other packages.

5. Build the Gluster Native Client software using the following commands:

```
# make

# make install
```

6. Verify that the correct version of Gluster Native Client is installed, using the following command:

```
# glusterfs –-version
```

6.1.2. Mounting Volumes

After installing the Gluster Native Client, you need to mount Gluster volumes to access data. There are two methods you can choose:

- *Section 6.1.2.1, “Manually Mounting Volumes”*
- *Section 6.1.2.2, “Automatically Mounting Volumes”*

After mounting a volume, you can test the mounted volume using the procedure described in *Section 6.4, “Testing Mounted Volumes”*.

Note

Server names selected during creation of Volumes should be resolvable in the client machine. You can use appropriate /etc/hosts entries or DNS server to resolve server names to IP addresses.

6.1.2.1. Manually Mounting Volumes

To manually mount a Gluster volume

- To mount a volume, use the following command:

```
# mount -t glusterfs HOSTNAME-OR-IPADDRESS:/VOLNAME MOUNTDIR
```

For example:

```
# mount -t glusterfs server1:/test-volume /mnt/glusterfs
```

Note

The server specified in the mount command is only used to fetch the gluster configuration volfile describing the volume name. Subsequently, the client will communicate directly with the servers mentioned in the volfile (which might not even include the one used for mount).

6.1.2.2. Automatically Mounting Volumes

To automatically mount a Gluster volume

- To mount a volume, edit the /etc/fstab file and add the following line:

  ```
  HOSTNAME-OR-IPADDRESS:/VOLNAME MOUNTDIR glusterfs defaults,_netdev 0 0
  ```

  For example:

  ```
  server1:/test-volume /mnt/glusterfs glusterfs defaults,_netdev 0 0
  ```

Mounting Options

You can specify the following options when using the **`mount -t glusterfs`** command. Note that you need to separate all options with commas.

backupvolfile-server=server-name

fetch-attempts=N (where N is number of attempts)

log-level=loglevel

log-file=logfile

direct-io-mode=[enable|disable]

ro (for readonly mounts)

acl (for enabling posix-ACLs)

worm (making the mount WORM - Write Once, Read Many type)

selinux (enable selinux on GlusterFS mount

For example:

```
# mount -t glusterfs -o backupvolfile-server=volfile_server2,fetch-attempts=2,log-level=WARNING,log-file=/var/log/gluster.log server1:/test-volume /mnt/glusterfs
```

Using /etc/fstab, options would look like below:

```
HOSTNAME-OR-IPADDRESS:/VOLNAME MOUNTDIR glusterfs defaults,_netdev,log-level=WARNING,log-file=/var/log/gluster.log 0 0
```

If **backupvolfile-server** option is added while mounting fuse client, when the first volfile server fails, then the server specified in **backupvolfile-server** option is used as volfile server to mount the client.

In **fetch-attempts=N** option, specify the number of attempts to fetch volume files while mounting a volume. This option will be useful when round-robin DNS is configured for the server-name.

6.2. NFS

You can use NFS v3 to access to gluster volumes.

GlusterFS 3.3.0, now includes network lock manager (NLM) v4 feature too. NLM enables applications on NFSv3 clients to do record locking on files. NLM program is started automatically with the NFS server process.

This section describes how to use NFS to mount Gluster volumes (both manually and automatically).

6.2.1. Using NFS to Mount Volumes

You can use either of the following methods to mount Gluster volumes:

- *Section 6.2.1.1, "Manually Mounting Volumes Using NFS "*
- *Section 6.2.1.2, "Automatically Mounting Volumes Using NFS"*

After mounting a volume, you can test the mounted volume using the procedure described in *Section 6.4, "Testing Mounted Volumes"*.

6.2.1.1. Manually Mounting Volumes Using NFS

To manually mount a Gluster volume using NFS

- To mount a volume, use the following command:

  ```
  # mount -t nfs -o vers=3 HOSTNAME-OR-IPADDRESS:/VOLNAME MOUNTDIR
  ```

  For example:

  ```
  # mount -t nfs -o vers=3 server1:/test-volume /mnt/glusterfs
  ```

Note

Gluster NFS server does not support UDP. If the NFS client you are using defaults to connecting using UDP, the following message appears:

requested NFS version or transport protocol is not supported.

To connect using TCP

- Add the following option to the mount command:

  ```
  -o mountproto=tcp
  ```

  For example:

  ```
  # mount -o mountproto=tcp,vers=3 -t nfs server1:/test-volume /mnt/
  glusterfs
  ```

To mount Gluster NFS server from a Solaris client

- Use the following command:

  ```
  # mount -o proto=tcp,vers=3 nfs://HOSTNAME-OR-IPADDRESS:38467/VOLNAME
  MOUNTDIR
  ```

  For example:

  ```
  # mount -o proto=tcp,vers=3 nfs://server1:38467/test-volume /mnt/
  glusterfs
  ```

6.2.1.2. Automatically Mounting Volumes Using NFS

You can configure your system to automatically mount Gluster volumes using NFS each time the system starts.

To automatically mount a Gluster volume using NFS

- To mount a volume, edit the /etc/fstab file and add the following line:

  ```
  HOSTNAME-OR-IPADDRESS:/VOLNAME MOUNTDIR nfs defaults,_netdev,vers=3 0 0
  ```

  For example,

  ```
  server1:/test-volume /mnt/glusterfs nfs defaults,_netdev,vers=3 0 0
  ```

  If default transport to mount NFS is UDP, use below line in fstab

  ```
  server1:/test-volume /mnt/glusterfs nfs defaults,_netdev,mountproto=tcp 0
  0
  ```

To automount NFS mounts

Gluster supports *nix standard method of automounting NFS mounts. Update the /etc/auto.master and /etc/auto.misc and restart the autofs service. After that, whenever a user or process attempts to access the directory it will be mounted in the background.

6.3. CIFS

You can use CIFS to access to volumes when using Microsoft Windows as well as SAMBA clients. For this access method, Samba packages need to be present on the client side. You can export glusterfs mount point as the samba export, and then mount it using CIFS protocol.

This section describes how to mount CIFS shares on Microsoft Windows-based clients (both manually and automatically) and how to verify that the volume has mounted successfully.

Note

CIFS access using the Mac OS X Finder is not supported, however, you can use the Mac OS X command line to access Gluster volumes using CIFS.

6.3.1. Using CIFS to Mount Volumes

You can use either of the following methods to mount Gluster volumes:

- *Section 6.3.1.2, "Manually Mounting Volumes Using CIFS "*
- *Section 6.3.1.3, "Automatically Mounting Volumes Using CIFS"*

After mounting a volume, you can test the mounted volume using the procedure described in *Section 6.4, "Testing Mounted Volumes"*.

You can also use Samba for exporting Gluster Volumes through CIFS protocol.

6.3.1.1. Exporting Gluster Volumes Through Samba

We recommend you to use Samba for exporting Gluster volumes through the CIFS protocol.

To export volumes through CIFS protocol

1. Mount a Gluster volume. For more information on mounting volumes, see *Section 6.1.2, "Mounting Volumes"*.
2. Setup Samba configuration to export the mount point of the Gluster volume.

   For example, if a Gluster volume is mounted on /mnt/gluster, you must edit smb.conf file to enable exporting this through CIFS. Open smb.conf file in an editor and add the following lines for a simple configuration:

   [glustertest]

   comment = For testing a Gluster volume exported through CIFS

   path = /mnt/glusterfs

   read only = no

   guest ok = yes

Save the changes and start the smb service using your systems init scripts (/etc/init.d/smb [re]start).

Note

To be able mount from any server in the trusted storage pool, you must repeat these steps on each Gluster node. For more advanced configurations, see Samba documentation.

6.3.1.2. Manually Mounting Volumes Using CIFS

You can manually mount Gluster volumes using CIFS on Microsoft Windows-based client machines.

To manually mount a Gluster volume using CIFS

1. Using Windows Explorer, choose **Tools > Map Network Drive…** from the menu. The **Map Network Drive** window appears.
2. Choose the drive letter using the **Drive** drop-down list.
3. Click **Browse**, select the volume to map to the network drive, and click **OK**.
4. Click **Finish.**

The network drive (mapped to the volume) appears in the Computer window.

Alternatively, to manually mount a Gluster volume using CIFS.

- Click **Start > Run** and enter the following:

  ```
  \\SERVERNAME\VOLNAME
  ```

  For example:

  ```
  \\server1\test-volume
  ```

6.3.1.3. Automatically Mounting Volumes Using CIFS

You can configure your system to automatically mount Gluster volumes using CIFS on Microsoft Windows-based clients each time the system starts.

To automatically mount a Gluster volume using CIFS

The network drive (mapped to the volume) appears in the Computer window and is reconnected each time the system starts.

1. Using Windows Explorer, choose **Tools > Map Network Drive…** from the menu. The **Map Network Drive** window appears.
2. Choose the drive letter using the **Drive** drop-down list.
3. Click **Browse**, select the volume to map to the network drive, and click **OK**.
4. Click the **Reconnect** at logon checkbox.
5. Click **Finish.**

6.4. Testing Mounted Volumes

To test mounted volumes

- Use the following command:

  ```
  # mount
  ```

  If the gluster volume was successfully mounted, the output of the mount command on the client will be similar to this example:

  ```
  server1:/test-volume on /mnt/glusterfs type fuse.glusterfs
  (rw,allow_other,default_permissions,max_read=131072
  ```

- Use the following command:

  # df -h

  The output of df command on the client will display the aggregated storage space from all the bricks in a volume similar to this example:

  # df -h /mnt/glusterfs

  Filesystem Size Used Avail Use% Mounted on

  server1:/test-volume 28T 22T 5.4T 82% /mnt/glusterfs

- Change to the directory and list the contents by entering the following:

  # cd MOUNTDIR

  # ls

- For example,

  # cd /mnt/glusterfs

  # ls

Managing GlusterFS Volumes

This section describes how to perform common GlusterFS management operations, including the following:

7.1. Tuning Volume Options

You can tune volume options, as needed, while the cluster is online and available.

Note

It is recommend to set server.allow-insecure option to ON if there are too many bricks in each volume or if there are too many services which have already utilized all the privileged ports in the system. Turning this option ON allows ports to accept/reject messages from insecure ports. So, use this option only if your deployment requires it.

To tune volume options

- Tune volume options 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
  ```

  The following table lists the Volume options along with its description and default value:

Note

The default options given here are subject to modification at any given time and may not be the same for all versions.

Option	Description	Default Value	Available Options
auth.allow	IP addresses of the clients which should be allowed to access the volume.	* (allow all)	Valid IP address which includes wild card patterns including *, such as 192.168.1.*
auth.reject	IP addresses of the clients which should be denied to access the volume.	NONE (reject none)	Valid IP address which includes wild card patterns including *, such as 192.168.2.*
client.grace-timeout	Specifies the duration for the lock state to be maintained on the client after a network disconnection.	10	10 - 1800 secs
cluster.self-heal-window-size	Specifies the maximum number of blocks per file on which self-heal would happen simultaneously.	16	0 - 1025 blocks
cluster.data-self-heal-algorithm	Specifies the type of self-heal. If you set the option as "full", the entire file is copied from source to destinations. If the option is set to "diff" the file blocks that are not in sync are copied to destinations. Reset uses a heuristic model. If the file does not exist on one of the subvolumes, or a zero-byte file exists (created by entry self-heal) the entire content has to be copied anyway, so there is no benefit from using the "diff" algorithm. If the file size is about the same	reset	full | diff | reset

Option	Description	Default Value	Available Options
	as page size, the entire file can be read and written with a few operations, which will be faster than "diff" which has to read checksums and then read and write.		
cluster.min-free-disk	Specifies the percentage of disk space that must be kept free. Might be useful for non-uniform bricks.	10%	Percentage of required minimum free disk space
cluster.stripe-block-size	Specifies the size of the stripe unit that will be read from or written to.	128 KB (for all files)	size in bytes
cluster.self-heal-daemon	Allows you to turn-off proactive self-heal on replicated volumes.	on	On | Off
diagnostics.brick-log-level	Changes the log-level of the bricks.	INFO	DEBUG|WARNING|ERROR|CRITICAL|NONE|TRACE
diagnostics.client-log-level	Changes the log-level of the clients.	INFO	DEBUG|WARNING|ERROR|CRITICAL|NONE|TRACE
diagnostics.latency-measurement	Statistics related to the latency of each operation would be tracked.	off	On | Off
diagnostics.dump-fd-stats	Statistics related to file-operations would be tracked.	off	On | Off
feature.read-only	Enables you to mount the entire volume as read-only for all the clients (including NFS clients) accessing it.	off	On | Off
features.lock-heal	Enables self-healing of locks when the network disconnects.	on	On | Off
features.quota-timeout	For performance reasons, quota caches the directory sizes on client. You can set timeout indicating the maximum duration of directory sizes in	0	0 - 3600 secs

Option	Description	Default Value	Available Options
	cache, from the time they are populated, during which they are considered valid.		
geo-replication.indexing	Use this option to automatically sync the changes in the filesystem from Master to Slave.	off	On | Off
network.frame-timeout	The time frame after which the operation has to be declared as dead, if the server does not respond for a particular operation.	1800 (30 mins)	1800 secs
network.ping-timeout	The time duration for which the client waits to check if the server is responsive. When a ping timeout happens, there is a network disconnect between the client and server. All resources held by server on behalf of the client get cleaned up. When a reconnection happens, all resources will need to be re-acquired before the client can resume its operations on the server. Additionally, the locks will be acquired and the lock tables updated. This reconnect is a very expensive operation and should be avoided.	42 Secs	42 Secs
nfs.enable-ino32	For 32-bit nfs clients or applications that do not support 64-bit inode numbers or large files, use this option from the CLI to make Gluster NFS return 32-bit inode numbers instead of 64-bit inode numbers. Applications that will	off	On | Off

Option	Description	Default Value	Available Options
	benefit are those that were either: * Built 32-bit and run on 32-bit machines. * Built 32-bit on 64-bit systems. * Built 64-bit but use a library built 32-bit, especially relevant for python and perl scripts. Either of the conditions above can lead to application on Linux NFS clients failing with "Invalid argument" or "Value too large for defined data type" errors.		
nfs.volume-access	Set the access type for the specified sub-volume.	read-write	read-write|read-only
nfs.trusted-write	If there is an UNSTABLE write from the client, STABLE flag will be returned to force the client to not send a COMMIT request. In some environments, combined with a replicated GlusterFS setup, this option can improve write performance. This flag allows users to trust Gluster replication logic to sync data to the disks and recover when required. COMMIT requests if received will be handled in a default manner by fsyncing. STABLE writes are still handled in a sync manner.	off	On | Off
nfs.trusted-sync	All writes and COMMIT requests	off	On | Off

Option	Description	Default Value	Available Options
	are treated as async. This implies that no write requests are guaranteed to be on server disks when the write reply is received at the NFS client. Trusted sync includes trusted-write behavior.		
nfs.export-dir	By default, all sub-volumes of NFS are exported as individual exports. Now, this option allows you to export only the specified subdirectory or subdirectories in the volume. This option can also be used in conjunction with nfs3.export-volumes option to restrict exports only to the subdirectories specified through this option. You must provide an absolute path.	Enabled for all sub directories.	Enable | Disable
nfs.export-volumes	Enable/Disable exporting entire volumes, instead if used in conjunction with nfs3.export-dir, can allow setting up only subdirectories as exports.	on	On | Off
nfs.rpc-auth-unix	Enable/Disable the AUTH_UNIX authentication type. This option is enabled by default for better interoperability. However, you can disable it if required.	on	On | Off
nfs.rpc-auth-null	Enable/Disable the AUTH_NULL authentication type. It is not recommended to change the default value for this option.	on	On | Off

Option	Description	Default Value	Available Options
nfs.rpc-auth-allow<IP-Addresses>	Allow a comma separated list of addresses and/or hostnames to connect to the server. By default, all clients are disallowed. This allows you to define a general rule for all exported volumes.	Reject All	IP address or Host name
nfs.rpc-auth-reject IP-Addresses	Reject a comma separated list of addresses and/or hostnames from connecting to the server. By default, all connections are disallowed. This allows you to define a general rule for all exported volumes.	Reject All	IP address or Host name
nfs.ports-insecure	Allow client connections from unprivileged ports. By default only privileged ports are allowed. This is a global setting in case insecure ports are to be enabled for all exports using a single option.	off	On | Off
nfs.addr-namelookup	Turn-off name lookup for incoming client connections using this option. In some setups, the name server can take too long to reply to DNS queries resulting in timeouts of mount requests. Use this option to turn off name lookups during address authentication. Note, turning this off will prevent you from using hostnames in rpc-auth.addr.* filters.	on	On | Off
nfs.register-with-portmap	For systems that need to run multiple NFS	on	On | Off

Option	Description	Default Value	Available Options
	servers, you need to prevent more than one from registering with portmap service. Use this option to turn off portmap registration for Gluster NFS.		
nfs.port <PORT-NUMBER>	Use this option on systems that need Gluster NFS to be associated with a non-default port number.	38465- 38467	
nfs.disable	Turn-off volume being exported by NFS	off	On | Off
performance.write-behind-window-size	Size of the per-file write-behind buffer.	1 MB	Write-behind cache size
performance.io-thread-count	The number of threads in IO threads translator.	16	0 - 65
performance.flush-behind	If this option is set ON, instructs write-behind translator to perform flush in background, by returning success (or any errors, if any of previous writes were failed) to application even before flush is sent to backend filesystem.	On	On | Off
performance.cache-max-file-size	Sets the maximum file size cached by the io-cache translator. Can use the normal size descriptors of KB, MB, GB,TB or PB (for example, 6GB). Maximum size uint64.	2 ^ 64 -1 bytes	size in bytes
performance.cache-min-file-size	Sets the minimum file size cached by the io-cache translator. Values same as "max" above.	0B	size in bytes
performance.cache-refresh-timeout	The cached data for a file will be retained till 'cache-refresh-timeout' seconds, after which data re-validation is performed.	1 sec	0 - 61

Option	Description	Default Value	Available Options
performance.cache-size	Size of the read cache.	32 MB	size in bytes
server.allow-insecure	Allow client connections from unprivileged ports. By default only privileged ports are allowed. This is a global setting in case insecure ports are to be enabled for all exports using a single option.	on	On | Off
server.grace-timeout	Specifies the duration for the lock state to be maintained on the server after a network disconnection.	10	10 - 1800 secs
server.statedump-path	Location of the state dump file.	/tmp directory of the brick	New directory path

You can view the changed volume options using the **`# gluster volume info VOLNAME`** command. For more information, see *Section 7.7, "Deleting Volumes"*.

7.2. Expanding Volumes

You can expand volumes, as needed, while the cluster is online and available. For example, you might want to add a brick to a distributed volume, thereby increasing the distribution and adding to the capacity of the GlusterFS volume.

Similarly, you might want to add a group of bricks to a distributed replicated volume, increasing the capacity of the GlusterFS volume.

Note

When expanding distributed replicated and distributed striped volumes, you need to add a number of bricks that is a multiple of the replica or stripe 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, etc.).

To expand a volume

1. On the first server in the cluster, probe the server to which you want to add the new brick using the following command:

   `# gluster peer probe HOSTNAME`

   For example:

   ```
   # gluster peer probe server4
   ```

```
Probe successful
```

2. Add the brick using the following command:

   # gluster volume add-brick *VOLNAME NEW-BRICK*

   For example:

```
# gluster volume add-brick test-volume server4:/exp4
Add Brick successful
```

3. Check the volume information using the following command:

   # gluster volume info

   The command displays information similar to the following:

```
Volume Name: test-volume
Type: Distribute
Status: Started
Number of Bricks: 4
Bricks:
Brick1: server1:/exp1
Brick2: server2:/exp2
Brick3: server3:/exp3
Brick4: server4:/exp4
```

4. Rebalance the volume to ensure that all files are distributed to the new brick.

   You can use the rebalance command as described in *Section 7.5, "Rebalancing Volumes"*.

7.3. Shrinking Volumes

You can shrink volumes, as needed, while the cluster is online and available. For example, you might need to remove a brick that has become inaccessible in a distributed volume due to hardware or network failure.

Note

Data residing on the brick that you are removing will no longer be accessible at the Gluster mount point. Note however that only the configuration information is removed - you can continue to access the data directly from the brick, as necessary.

When shrinking distributed replicated and distributed striped 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 striped volume with a stripe count of 2, you need to remove bricks in multiples of 2 (such as 4, 6, 8, etc.). In addition, the bricks you are trying to remove must be from the same sub-volume (the same replica or stripe set).

To shrink a volume

1. Remove the brick using the following command:

 # gluster volume remove-brick VOLNAME *BRICK* start

For example, to remove server2:/exp2:

```
# gluster volume remove-brick test-volume server2:/exp2 start

Removing brick(s) can result in data loss. Do you want to Continue? (y/n)
```

2. Enter "y" to confirm the operation. The command displays the following message indicating that the remove brick operation is successfully started:

```
Remove Brick successful
```

3. (Optional) View the status of the remove brick operation using the following command:

 # gluster volume remove-brick VOLNAME *BRICK* status

For example, to view the status of remove brick operation on server2:/exp2 brick:

```
# gluster volume remove-brick test-volume server2:/exp2 status
                                    Node  Rebalanced-files  size  scanned       status
                               ---------  ----------------  ----  -------  -----------
617c923e-6450-4065-8e33-865e28d9428f               34   340      162   in progress
```

4. Commit the remove brick operation using the following command:

 # gluster volume remove-brick VOLNAME *BRICK* commit

For example, to view the status of remove brick operation on server2:/exp2 brick:

```
# gluster volume remove-brick test-volume server2:/exp2 commit
```

```
Remove Brick successful
```

5. 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:/exp1
Brick3: server3:/exp3
Brick4: server4:/exp4
```

6. Rebalance the volume to ensure that all files are distributed to the new brick.

You can use the rebalance command as described in *Section 7.5, "Rebalancing Volumes"*.

7.4. Migrating Volumes

You can migrate the data from one brick to another, as needed, while the cluster is online and available.

To migrate a volume

1. Make sure the new brick, server5 in this example, is successfully added to the cluster.

   For more information, see *Section 4.1, "Adding Servers to Trusted Storage Pool"*.

2. Migrate the data from one brick to another using the following command:

   `# gluster volume replace-brick VOLNAME BRICKNEW-BRICK start`

   For example, to migrate the data in server3:/exp3 to server5:/exp5 in test-volume:

   ```
   # gluster volume replace-brick test-volume server3:/exp3  server5:exp5 start
   Replace brick start operation successful
   ```

   Note

   You need to have the FUSE package installed on the server on which you are running the replace-brick command for the command to work.

3. To pause the migration operation, if needed, use the following command:

   `# gluster volume replace-brick VOLNAME BRICK NEW-BRICK  pause`

   For example, to pause the data migration from server3:/exp3 to server5:/exp5 in test-volume:

   ```
   # gluster volume replace-brick test-volume server3:/exp3 server5:exp5 pause
   Replace brick pause operation successful
   ```

4. To abort the migration operation, if needed, use the following command:

   `# gluster volume replace-brick VOLNAME BRICK NEW-BRICK abort`

   For example, to abort the data migration from server3:/exp3 to server5:/exp5 in test-volume:

   ```
   # gluster volume replace-brick test-volume server3:/exp3 server5:exp5 abort
   Replace brick abort operation successful
   ```

5. Check the status of the migration operation using the following command:

   `# gluster volume replace-brick VOLNAME BRICK NEW-BRICK status`

   For example, to check the data migration status from server3:/exp3 to server5:/exp5 in test-volume:

   ```
   # gluster volume replace-brick test-volume server3:/exp3 server5:/exp5 status
   Current File = /usr/src/linux-headers-2.6.31-14/block/Makefile
   ```

```
Number of files migrated = 10567
Migration complete
```

The status command shows the current file being migrated along with the current total number of files migrated. After completion of migration, it displays Migration complete.

6. Commit the migration of data from one brick to another using the following command:

   # gluster volume replace-brick VOLNAME BRICK NEW-BRICK commit

   For example, to commit the data migration from server3:/exp3 to server5:/exp5 in test-volume:

```
# gluster volume replace-brick test-volume server3:/exp3 server5:/exp5 commit
replace-brick commit successful
```

7. Verify the migration of brick by viewing the volume info using the following command:

   # gluster volume info VOLNAME

   For example, to check the volume information of new brick server5:/exp5 in test-volume:

```
# gluster volume info test-volume
Volume Name: testvolume
Type: Replicate
Status: Started
Number of Bricks: 4
Transport-type: tcp
Bricks:
Brick1: server1:/exp1
Brick2: server2:/exp2
Brick3: server4:/exp4
Brick4: server5:/exp5

The new volume details are displayed.
```

   The new volume details are displayed.

   In the above example, previously, there were bricks; 1,2,3, and 4 and now brick 3 is replaced by brick 5.

7.5. Rebalancing Volumes

After expanding or shrinking a volume (using the add-brick and remove-brick commands respectively), you need to rebalance the data among the servers. New directories created after expanding or shrinking of the volume will be evenly distributed automatically. For all the existing directories, the distribution can be fixed by rebalancing the layout and/or data.

This section describes how to rebalance GlusterFS volumes in your storage environment, using the following common scenarios:

- Fix Layout - Fixes the layout changes so that the files can actually go to newly added nodes. For more information, see *Section 7.5.1, “Rebalancing Volume to Fix Layout Changes”*.
- Fix Layout and Migrate Data - Rebalances volume by fixing the layout changes and migrating the existing data. For more information, see *Section 7.5.2, “Rebalancing Volume to Fix Layout and Migrate Data”*.

7.5.1. Rebalancing Volume to Fix Layout Changes

Fixing the layout is necessary because the layout structure is static for a given directory. In a scenario where new bricks have been added to the existing volume, newly created files in existing directories will still be distributed only among the old bricks. The **# gluster volume rebalance VOLNAME fix-layout start** command will fix the layout information so that the files can also go to newly added nodes. When this command is issued, all the file stat information which is already cached will get revalidated.

A fix-layout rebalance will only fix the layout changes and does not migrate data. If you want to migrate the existing data, use**# gluster volume rebalance VOLNAME start** command to rebalance data among the servers.

To rebalance a volume to fix layout changes

- Start the rebalance operation on any one of the server using the following command:

  # gluster volume rebalance VOLNAME fix-layout start

  For example:

```
# gluster volume rebalance test-volume fix-layout start
Starting rebalance on volume test-volume has been successful
```

7.5.2. Rebalancing Volume to Fix Layout and Migrate Data

After expanding or shrinking a volume (using the add-brick and remove-brick commands respectively), you need to rebalance the data among the servers.

To rebalance a volume to fix layout and migrate the existing data

- Start the rebalance operation on any one of the server using the following command:

  # gluster volume rebalance VOLNAME start

  For example:

```
# gluster volume rebalance test-volume start
Starting rebalancing on volume test-volume has been successful
```

- Start the migration operation forcefully on any one of the server using the following command:

  # gluster volume rebalance VOLNAME start force

  For example:

```
# gluster volume rebalance test-volume start force
Starting rebalancing on volume test-volume has been successful
```

7.5.3. Displaying Status of Rebalance Operation

You can display the status information about rebalance volume operation, as needed.

To view status of rebalance volume

- Check the status of the rebalance operation, using the following command:

  # gluster volume rebalance *VOLNAME* status

For example:

```
# gluster volume rebalance test-volume status
                                Node  Rebalanced-files  size  scanned       status
                           ---------  ----------------  ----  -------  -----------
617c923e-6450-4065-8e33-865e28d9428f               416  1463      312  in progress
```

The time to complete the rebalance operation depends on the number of files on the volume along with the corresponding file sizes. Continue checking the rebalance status, verifying that the number of files rebalanced or total files scanned keeps increasing.

For example, running the status command again might display a result similar to the following:

```
# gluster volume rebalance test-volume status
                                Node  Rebalanced-files  size  scanned       status
                           ---------  ----------------  ----  -------  -----------
617c923e-6450-4065-8e33-865e28d9428f               498  1783      378  in progress
```

The rebalance status displays the following when the rebalance is complete:

```
# gluster volume rebalance test-volume status
                                Node  Rebalanced-files  size  scanned       status
                           ---------  ----------------  ----  -------  -----------
617c923e-6450-4065-8e33-865e28d9428f               502  1873      334   completed
```

7.5.4. Stopping Rebalance Operation

You can stop the rebalance operation, as needed.

To stop rebalance

- Stop the rebalance operation using the following command:

  # gluster volume rebalance *VOLNAME* stop

  For example:

```
# gluster volume rebalance test-volume stop
                                Node  Rebalanced-files  size  scanned       status
                           ---------  ----------------  ----  -------  -----------
617c923e-6450-4065-8e33-865e28d9428f                59   590      244       stopped
Stopped rebalance process on volume test-volume
```

7.6. Stopping Volumes

To stop a volume

1. Stop the volume using 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)
```

2. Enter **y** to confirm the operation. The output of the command displays the following:

```
Stopping volume test-volume has been successful
```

7.7. Deleting Volumes

To delete a volume

1. Delete the volume using 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)
```

2. Enter **y** to confirm the operation. The command displays the following:

```
Deleting volume test-volume has been successful
```

7.8. Triggering Self-Heal on Replicate

In replicate module, previously you had to manually trigger a self-heal when a brick goes offline and comes back online, to bring all the replicas in sync. Now the pro-active self-heal daemon runs in the background, diagnoses issues and automatically initiates self-healing every 10 minutes on the files which requires *healing*.

You can view the list of files that need *healing*, the list of files which are currently/previously *healed*, list of files which are in split-brain state, and you can manually trigger self-heal on the entire volume or only on the files which need *healing*.

- Trigger self-heal only on the files which requires *healing*:

  # gluster volume heal *VOLNAME*

  For example, to trigger self-heal on files which requires *healing* of test-volume:

```
# gluster volume heal test-volume
Heal operation on volume test-volume has been successful
```

- Trigger self-heal on all the files of a volume:

  # gluster volume heal *VOLNAME* full

  For example, to trigger self-heal on all the files of of test-volume:

```
# gluster volume heal test-volume full
Heal operation on volume test-volume has been successful
```

- View the list of files that needs *healing*:

  # gluster volume heal *VOLNAME* info

For example, to view the list of files on test-volume that needs *healing*:

```
# gluster volume heal test-volume info
Brick server1:/gfs/test-volume_0
Number of entries: 0

Brick server2:/gfs/test-volume_1
Number of entries: 101
/95.txt
/32.txt
/66.txt
/35.txt
/18.txt
/26.txt
/47.txt
/55.txt
/85.txt
...
```

- View the list of files that are self-healed:

  # gluster volume heal *VOLNAME* info healed

  For example, to view the list of files on test-volume that are self-healed:

```
# gluster volume heal test-volume info healed
Brick server1:/gfs/test-volume_0
Number of entries: 0

Brick server2:/gfs/test-volume_1
Number of entries: 69
/99.txt
/93.txt
/76.txt
/11.txt
/27.txt
/64.txt
/80.txt
/19.txt
/41.txt
/29.txt
/37.txt
/46.txt
...
```

- View the list of files of a particular volume on which the self-heal failed:

  # gluster volume heal *VOLNAME* info failed

  For example, to view the list of files of test-volume that are not self-healed:

```
# gluster volume heal test-volume info failed
Brick server1:/gfs/test-volume_0
Number of entries: 0

Brick server2:/gfs/test-volume_3
Number of entries: 72
/90.txt
/95.txt
/77.txt
/71.txt
```

```
/87.txt
/24.txt
...
```

- View the list of files of a particular volume which are in split-brain state:

  # gluster volume heal *VOLNAME* info split-brain

  For example, to view the list of files of test-volume which are in split-brain state:

```
# gluster volume heal test-volume info split-brain
Brick server1:/gfs/test-volume_2
Number of entries: 12
/83.txt
/28.txt
/69.txt
...

Brick server2:/gfs/test-volume_2
Number of entries: 12
/83.txt
/28.txt
/69.txt
...
```

Managing Geo-replication

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

Geo-replication uses a master–slave model, whereby replication and mirroring occurs between the following partners:

- Master – a GlusterFS volume
- Slave – a slave which can be of the following types:
  - A local directory which can be represented as file URL like **`file:///path/to/dir`**. You can use shortened form, for example, **`/path/to/dir`**.
  - A GlusterFS Volume - Slave volume can be either a local volume like **`gluster://localhost:volname`** (shortened form - **`:volname`**) or a volume served by different host like **`gluster://host:volname`** (shortened form - **`host:volname`**).

Note

Both of the above types can be accessed remotely using SSH tunnel. To use SSH, add an SSH prefix to either a file URL or gluster type URL. For example, **`ssh://root@remote-host:/path/to/dir`** (shortened form - **`root@remote-host:/path/to/dir`**) or **`ssh://root@remote-host:gluster://localhost:volname`** (shortened from - **`root@remote-host::volname`**).

This section introduces Geo-replication, illustrates the various deployment scenarios, and explains how to configure the system to provide replication and mirroring in your environment.

8.1. Replicated Volumes vs Geo-replication

The following table lists the difference between replicated volumes and geo-replication:

Replicated Volumes	Geo-replication
Mirrors data across nodes in a cluster	Mirrors data across geographically distributed clusters
Provides high-availability	Ensures backing up of data for disaster recovery
Synchronous replication (each and every file modify operation is sent across all the bricks)	Asynchronous replication (checks for the changes in files periodically and syncs them on detecting differences)

8.2. Preparing to Deploy Geo-replication

This section provides an overview of the Geo-replication deployment scenarios, describes how you can check the minimum system requirements, and explores common deployment scenarios.

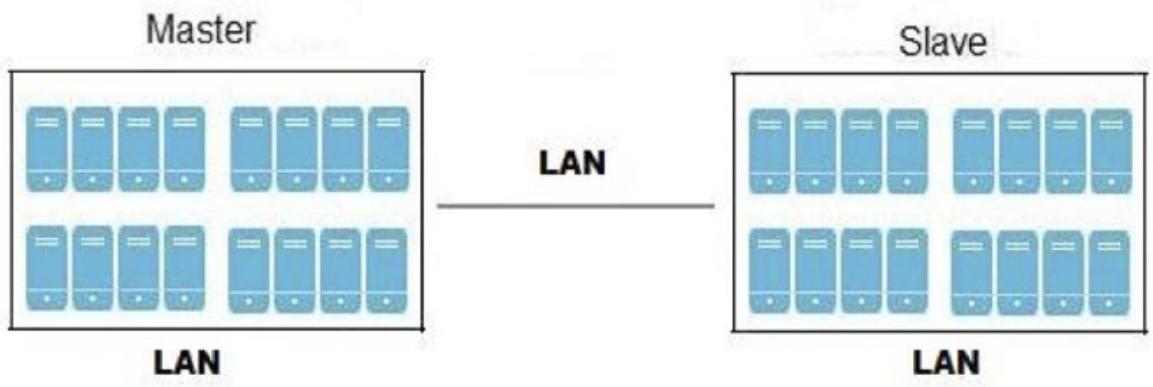
8.2.1. Exploring Geo-replication Deployment Scenarios

Geo-replication provides an incremental replication service over Local Area Networks (LANs), Wide Area Network (WANs), and across 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

You can configure Geo-replication to mirror data over a Local Area Network.



Geo-replication over WAN

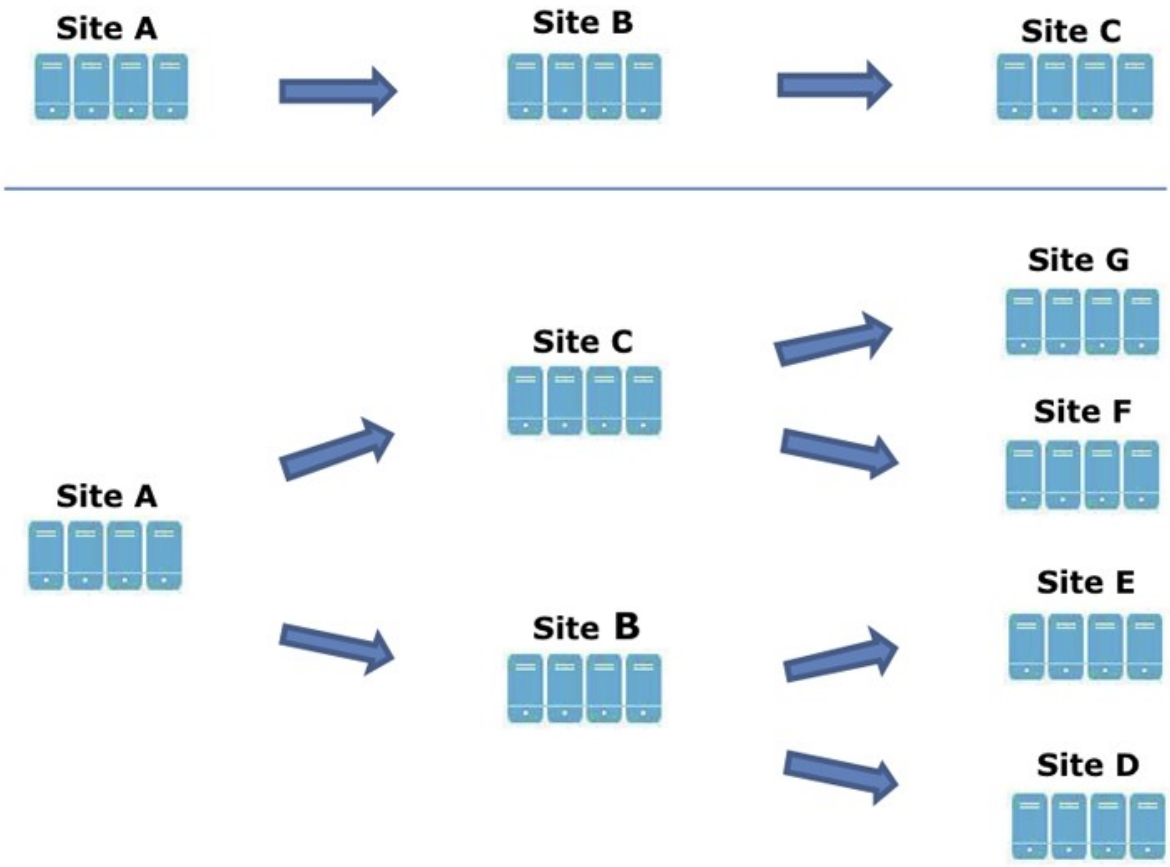
You can configure Geo-replication to replicate data over a Wide Area Network.

Geo-replication over Internet

You can configure Geo-replication to mirror data over the Internet.

Multi-site cascading Geo-replication

You can configure Geo-replication to mirror data in a cascading fashion across multiple sites.



8.2.2. Geo-replication Deployment Overview

Deploying Geo-replication involves the following steps:

1. Verify that your environment matches the minimum system requirements. For more information, see *Section 8.2.3, "Checking Geo-replication Minimum Requirements"*.
2. Determine the appropriate deployment scenario. For more information, see *Section 8.2.1, "Exploring Geo-replication Deployment Scenarios"*.
3. Start Geo-replication on master and slave systems, as required. For more information, see *Section 8.3, "Starting Geo-replication"*.

8.2.3. Checking Geo-replication Minimum Requirements

Before deploying GlusterFS Geo-replication, verify that your systems match the minimum requirements.

The following table outlines the minimum requirements for both master and slave nodes within your environment:

Component	Master	Slave
Operating System	GNU/Linux	GNU/Linux
Filesystem	GlusterFS 3.2 or higher	GlusterFS 3.2 or higher, ext3, ext4, or XFS (any other POSIX compliant file system would work, but has not been tested extensively)
Python	Python 2.4 (with ctypes external module), or Python 2.5 (or higher)	Python 2.4 (with ctypes external module), or Python 2.5 (or higher)
Secure shell	OpenSSH version 4.0 (or higher)	SSH2-compliant daemon
Remote synchronization	rsync 3.0.0 or higher	rsync 3.0.0 or higher
FUSE	GlusterFS supported versions	GlusterFS supported versions

8.2.4. Setting Up the Environment for Geo-replication

Time Synchronization

- All servers that are part of a geo-replication master volume need to have their clocks in sync. You are recommended to set up NTP (Network Time Protocol) daemon service to keep the clocks in sync.

  For example: In a Replicated volume where brick1 of the master is at 12.20 hrs and brick 2 of the master is at 12.10 hrs with 10 minutes time lag, all the changes in brick2 between this period may go unnoticed during synchronization of files with Slave.

  For more information on setting up NTP daemon, see *http://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/6/html/Migration_Planning_Guide/ch04s07.html*.

To setup Geo-replication for SSH

Password-less login has to be set up between the host machine (where geo-replication start command will be issued) and the remote machine (where slave process should be launched through SSH).

1. On the node where geo-replication start commands are to be issued, run the following command:

   ```
   # ssh-keygen -f /var/lib/glusterd/geo-replication/secret.pem
   ```

   Press Enter twice to avoid passphrase.

2. Run the following command on master for all the slave hosts:

   ```
   # ssh-copy-id -i /var/lib/glusterd/geo-replication/secret.pem.pub
   user@slavehost
   ```

8.2.5. Setting Up the Environment for a Secure Geo-replication Slave

You can configure a secure slave using SSH so that master is granted restricted access. With GlusterFS 3.3, you need not specify slave configuration parameters on the master-side. For example, the master does not require the location of the rsync program on slave but the slave must ensure that rsync is in the PATH of the user which the master connects using SSH. The only information that master and slave have to negotiate are the slave-side user account, slave's resources and the master's public key. Secure access to the slave can be established using the following options:

- Restricting Remote Command Execution
- Using **`Mountbroker`** for Slaves
- Using IP based Access Control

Backward Compatibility

Your existing Geo-replication environment will work with GlusterFS 3.3, except for the following:

- The process of secure reconfiguration affects only the GlusterFS instance on slave. The changes are transparent to master with the exception that you may have to change the SSH target to an unprivileged account on the slave.
- The following are some exceptions where backward compatibility cannot be provided:
  - Geo-replication URLs which specify the slave resource include the following special characters: space, *, ?, [;
  - Slave does not have glusterd running.

8.2.5.1. Restricting Remote Command Execution

If you restrict remote command execution, then the slave audits commands coming from the master and only the pre-configured commands are allowed. The slave also provides access only to the files which are pre-configured to be read or manipulated by the master.

To restrict remote command execution:

1. Identify the location of the gsyncd helper utility on Slave. This utility is installed in **`PREFIX/libexec/glusterfs/gsyncd`**, where PREFIX is a compile-time parameter of glusterfs. For example, **`--prefix=PREFIX`** to the configure script with the following common values **`/usr, /usr/local, and /opt/glusterfs/glusterfs_version`**.
2. Ensure that command invoked from master to slave is passed through the slave's gsyncd utility.

   You can use either of the following two options:

   - Set gsyncd with an absolute path as the shell for the account which the master connects through SSH. If you need to use a privileged account, then set it up by creating a new user with UID 0.
   - Setup key authentication with command enforcement to gsyncd. You must prefix the copy of master's public key in the Slave account's **`authorized_keys`** file with the following command:

     `command=<path to gsyncd>`.

     For example, **`command="PREFIX/glusterfs/gsyncd" ssh-rsa AAAAB3Nza....`**

8.2.5.2. Using Mountbroker for Slaves

mountbroker is a new service of glusterd. This service allows an unprivileged process to own a GlusterFS mount. This is accomplished by registering a label (and DSL (Domain-specific language) options) with glusterd through the glusterd volfile. Using CLI, you can send a mount request to glusterd and receive an alias (symlink) to the mounted volume.

The unprivileged process/agent uses the mountbroker service of glusterd to set up an auxiliary gluster mount. The mount is setup so as to allow only that agent to provide administrative level access to the particular volume.

To setup an auxiliary gluster mount for the agent:

1. Create a new group. For example, **geogroup**.
2. Create a unprivileged account. For example, **geoaccount**. Make it a member of **geogroup**.
3. Create a new directory as superuser to be used as mountbroker's root.
4. Change the permission of the directory to *0711.*
5. Add the following options to the glusterd volfile, located at /etc/glusterfs/glusterd.vol, assuming the name of the slave gluster volume as **slavevol**:

   option mountbroker-root /var/mountbroker-root

   option mountbroker-geo-replication.geoaccount slavevol

   option geo-replication-log-group geogroup

   A sample glusterd volfile along with default options:

   ```
   volume management
       type mgmt/glusterd
       option working-directory /etc/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 mountbroker-root /var/mountbroker-root
       option mountbroker-geo-replication.geoaccount slavevol
       option geo-replication-log-group geogroup
   end-volume
   ```

   If you host multiple slave volumes, you can repeat step 2. for each of the slave volumes and add the following options to the **volfile**:

   ```
   option mountbroker-geo-replication.geoaccount2 slavevol2
   option mountbroker-geo-replication.geoaccount3 slavevol3
   ```

6. Setup Master to access Slave as **geoaccount@Slave**.

   You can add multiple slave volumes within the same account (geoaccount) by providing comma-separated list of slave volumes (without spaces) as the argument of **mountbroker-geo-replication.geogroup**. You can also have multiple options of the form **mountbroker-geo-replication.*** . It is recommended to use one service account per Master machine. For example, if there are multiple slave volumes on Slave for the master machines Master1, Master2, and Master3, then create a dedicated service user on Slave for them by repeating Step 2. for each

(like geogroup1, geogroup2, and geogroup3), and then add the following corresponding options to the volfile:

```
option mountbroker-geo-replication.geoaccount1
slavevol11,slavevol12,slavevol13

option mountbroker-geo-replication.geoaccount2 slavevol21,slavevol22

option mountbroker-geo-replication.geoaccount3 slavevol31
```

Now set up Master1 to ssh to geoaccount1@Slave, etc.

You must restart glusterd to make the configuration changes effective.

8.2.5.3. Using IP based Access Control

You can provide access control for the slave resources using IP addresses. You can use method for both Gluster volume and and file tree slaves, but in this section, we are focusing on file tree slaves.

To set IP address based access control for file tree slaves:

1. Set a general restriction for accessibility of file tree resources:

   ```
   # gluster volume geo-replication '/*' config allow-network ::1,127.0.0.1
   ```

   This will refuse all requests for spawning slave agents except for requests initiated locally.

2. If you want the to lease file tree at **/data/slave-tree** to Master, enter the following command:

   ```
   # gluster volume geo-replication /data/slave-tree config allow-network
   MasterIP
   ```

   `MasterIP` is the IP address of Master. The slave agent spawn request from master will be accepted if it is executed at **/data/slave-tree**.

If the Master side network configuration does not enable the Slave to recognize the exact IP address of Master, you can use CIDR notation to specify a subnet instead of a single IP address as MasterIP or even comma-separated lists of CIDR subnets.

If you want to extend IP based access control to gluster slaves, use the following command:

```
# gluster volume geo-replication '*' config allow-network ::1,127.0.0.1
```

8.3. Starting Geo-replication

This section describes how to configure and start Gluster Geo-replication in your storage environment, and verify that it is functioning correctly.

8.3.1. Starting Geo-replication

To start Gluster Geo-replication

- Start geo-replication between the hosts using the following command:

  # gluster volume geo-replication *MASTER SLAVE* start

  For example:

```
# gluster volume geo-replication Volume1 example.com:/data/remote_dir start
Starting geo-replication session between Volume1
example.com:/data/remote_dir has been successful
```

Note

You may need to configure the Geo-replication service before starting it. For more information, see *Section 8.3.4, "Configuring Geo-replication"*.

8.3.2. Verifying Successful Deployment

You can use the gluster command to verify the status of Gluster Geo-replication in your environment.

To verify the status Gluster Geo-replication

- Verify the status by issuing the following command on host:

  # gluster volume geo-replication *MASTER SLAVE* status

  For example:

  # gluster volume geo-replication Volume1 example.com:/data/remote_dir status

```
# gluster volume geo-replication Volume1 example.com:/data/remote_dir status

MASTER    SLAVE                              STATUS
______    ______________________________     ____________
Volume1 root@example.com:/data/remote_dir  Starting....
```

8.3.3. Displaying Geo-replication Status Information

You can display status information about a specific geo-replication master session, or a particular master-slave session, or all geo-replication sessions, as needed.

To display geo-replication status information

- Display information of all geo-replication sessions using the following command:

```
# gluster volume geo-replication Volume1 example.com:/data/remote_dir status

MASTER    SLAVE                              STATUS
______    ______________________________     ____________
```

```
Volume1 root@example.com:/data/remote_dir  Starting....
```

- Display information of a particular master slave session using the following command:

  # gluster volume geo-replication *MASTER SLAVE* status

  For example, to display information of Volume1 and example.com:/data/remote_dir

  # gluster volume geo-replication Volume1 example.com:/data/remote_dir status

  The status of the geo-replication between Volume1 and example.com:/data/remote_dir is displayed.

- Display information of all geo-replication sessions belonging to a master

  # gluster volume geo-replication MASTER status

  For example, to display information of Volume1

```
# gluster volume geo-replication Volume1 example.com:/data/remote_dir status

MASTER    SLAVE                                 STATUS
_____    ____________________________________  ____________
Volume1 ssh://example.com:gluster://127.0.0.1:remove_volume  OK

Volume1 ssh://example.com:file:///data/remote_dir  OK
```

  The status of a session could be one of the following four:

- **Starting**: This is the initial phase of the Geo-replication session; it remains in this state for a minute, to make sure no abnormalities are present.
- **OK**: The geo-replication session is in a stable state.
- **Faulty**: The geo-replication session has witnessed some abnormality and the situation has to be investigated further. For further information, see *Chapter 14, Troubleshooting GlusterFS* section.
- **Corrupt**: The monitor thread which is monitoring the geo-replication session has died. This situation should not occur normally, if it persists contact Red Hat Support*www.redhat.com/support/*.

8.3.4. Configuring Geo-replication

To configure Gluster Geo-replication

- Use the following command at the Gluster command line:

  # gluster volume geo-replication *MASTER SLAVE* config [options]

  For more information about the options, see *Chapter 15, Command Reference* .

  For example:

  To view list of all option/value pair, use the following command:

  # gluster volume geo-replication Volume1 example.com:/data/remote_dir config

8.3.5. Stopping Geo-replication

You can use the gluster command to stop Gluster Geo-replication (syncing of data from Master to Slave) in your environment.

To stop Gluster Geo-replication

- Stop geo-replication between the hosts using the following command:

  `# gluster volume geo-replication MASTER SLAVE stop`

  For example:

  ```
  # gluster volume geo-replication Volume1 example.com:/data/remote_dir stop
  Stopping geo-replication session between Volume1 and
  example.com:/data/remote_dir has been successful
  ```

  See *Chapter 15, Command Reference* for more information about the gluster command.

8.4. Restoring Data from the Slave

You can restore data from the slave to the master volume, whenever the master volume becomes faulty for reasons like hardware failure.

The example in this section assumes that you are using the Master Volume (Volume1) with the following configuration:

```
machine1# gluster volume info
Type: Distribute
Status: Started
Number of Bricks: 2
Transport-type: tcp
Bricks:
Brick1: machine1:/export/dir16
Brick2: machine2:/export/dir16
Options Reconfigured:
geo-replication.indexing: on
```

The data is syncing from master volume (Volume1) to slave directory (example.com:/data/remote_dir). To view the status of this geo-replication session run the following command on Master:

```
# gluster volume geo-replication Volume1 root@example.com:/data/remote_dir status

MASTER    SLAVE                                STATUS
______    ______________________________    ____________
Volume1  root@example.com:/data/remote_dir   OK
```

Before Failure

Assume that the Master volume had 100 files and was mounted at /mnt/gluster on one of the client machines (client). Run the following command on Client machine to view the list of files:

```
client# ls /mnt/gluster | wc –l
100
```

The slave directory (example.com) will have same data as in the master volume and same can be viewed by running the following command on slave:

```
example.com# ls /data/remote_dir/ | wc –l
100
```

After Failure

If one of the bricks (machine2) fails, then the status of Geo-replication session is changed from "OK" to "Faulty". To view the status of this geo-replication session run the following command on Master:

```
# gluster volume geo-replication Volume1 root@example.com:/data/remote_dir status

MASTER    SLAVE                                STATUS
______    ______________________________       ____________
Volume1   root@example.com:/data/remote_dir  Faulty
```

Machine2 is failed and now you can see discrepancy in number of files between master and slave. Few files will be missing from the master volume but they will be available only on slave as shown below.

Run the following command on Client:

```
client # ls /mnt/gluster | wc –l
52
```

Run the following command on slave (example.com):

```
Example.com# # ls /data/remote_dir/ | wc –l
100
```

To restore data from the slave machine

1. Stop all Master's geo-replication sessions using the following command:

   # gluster volume geo-replication *MASTER SLAVE* stop

   For example:

   ```
   machine1# gluster volume geo-replication Volume1
   example.com:/data/remote_dir stop

   Stopping geo-replication session between Volume1 &
   example.com:/data/remote_dir has been successful
   ```

   Note

   Repeat **# gluster volume geo-replication *MASTER SLAVE* stop** command on all active geo-replication sessions of master volume.

2. Replace the faulty brick in the master by using the following command:

   # gluster volume replace-brick *VOLNAME BRICK NEW-BRICK* start

   For example:

```
machine1# gluster volume replace-brick Volume1 machine2:/export/dir16 machine3:/export/
dir16 start
Replace-brick started successfully
```

3. Commit the migration of data using the following command:

   # gluster volume replace-brick *VOLNAME BRICK NEW-BRICK* commit force

   For example:

```
machine1# gluster volume replace-brick Volume1 machine2:/export/dir16 machine3:/export/
dir16 commit force
Replace-brick commit successful
```

4. Verify the migration of brick by viewing the volume info using the following command:

   # gluster volume info *VOLNAME*

   For example:

```
machine1# gluster volume info
Volume Name: Volume1
Type: Distribute
Status: Started
Number of Bricks: 2
Transport-type: tcp
Bricks:
Brick1: machine1:/export/dir16
Brick2: machine3:/export/dir16
Options Reconfigured:
geo-replication.indexing: on
```

5. Run rsync command manually to sync data from slave to master volume's client (mount point).

   For example:

   example.com# rsync -PavhS --xattrs --ignore-existing /data/remote_dir/ client:/mnt/gluster

   Verify that the data is synced by using the following command:

   On master volume, run the following command:

```
Client # ls | wc –l
100
```

   On the Slave run the following command:

```
example.com# ls /data/remote_dir/ | wc –l
100
```

   Now Master volume and Slave directory is synced.

6. Restart geo-replication session from master to slave using the following command:

   # gluster volume geo-replication *MASTER SLAVE* start

For example:

```
machine1# gluster volume geo-replication Volume1
example.com:/data/remote_dir start
Starting geo-replication session between Volume1 &
example.com:/data/remote_dir has been successful
```

8.5. Best Practices

Manually Setting Time

If you have to change the time on your bricks manually, then you must set uniform time on all bricks. This avoids the out-of-time sync issue described in *Section 8.2.4, "Setting Up the Environment for Geo-replication"*. Setting time backward corrupts the geo-replication index, so the recommended way to set the time manually is:

1. Stop geo-replication between the master and slave using the following command:

   # gluster volume geo-replication *MASTER SLAVE* stop

2. Stop the geo-replication indexing using the following command:

   # gluster volume set *MASTER* geo-replication.indexing off

3. Set uniform time on all bricks.s

4. Restart your geo-replication sessions by using the following command:

   # gluster volume geo-replication *MASTER SLAVE* start

Running Geo-replication commands in one system

It is advisable to run the geo-replication commands in one of the bricks in the trusted storage pool. This is because, the log files for the geo-replication session would be stored in the *Server* where the Geo-replication start is initiated. Hence it would be easier to locate the log-files when required.

Isolation

Geo-replication slave operation is not sandboxed as of now and is ran as a privileged service. So for the security reason, it is advised to create a sandbox environment (dedicated machine / dedicated virtual machine / chroot/container type solution) by the administrator to run the geo-replication slave in it. Enhancement in this regard will be available in follow-up minor release.

Managing Directory Quota

Directory quotas in GlusterFS allow you to set limits on usage of disk space of a given directory. The storage administrators can control the disk space utilization at the directory level in GlusterFS by setting quota limits on the given directory. If admin sets the quota limit on the '/' of the volume, it can be treated as 'volume level quota'. GlusterFS's quota implementation can have different quota limit set on any directory and it can be nested. This is particularly useful in cloud deployments to facilitate utility billing model.

Note

For now, only Hard limit is supported. Here, the limit cannot be exceeded and attempts to use more disk space beyond the set limit will be denied.

System administrators can also monitor the resource utilization to limit the storage for the users depending on their role in the organization.

You can set the quota at the following levels:

- Directory level – limits the usage at the directory level
- Volume level – limits the usage at the volume level

Note

You can set the disk limit on the directory even if it is not created. The disk limit is enforced immediately after creating that directory. For more information on setting disk limit, see *Section 9.3, “Setting or Replacing Disk Limit ”*.

9.1. Enabling Quota

You must enable Quota to set disk limits.

To enable quota

- Enable the quota using the following command:

  `# gluster volume quota VOLNAME enable`

  For example, to enable quota on test-volume:

```
# gluster volume quota test-volume enable
Quota is enabled on /test-volume
```

9.2. Disabling Quota

You can disable Quota, if needed.

To disable quota:

- Disable the quota using the following command:

  # gluster volume quota *VOLNAME* disable

  For example, to disable quota on test-volume:

```
# gluster volume quota test-volume disable
Quota translator is disabled on /test-volume
```

9.3. Setting or Replacing Disk Limit

You can create new directories in your storage environment and set the disk limit or set disk limit for the existing directories. The directory name should be relative to the volume with the export directory/ mount being treated as "/".

To set or replace disk limit

- Set the disk limit using the following command:

  # gluster volume quota *VOLNAME* limit-usage */directorylimit-value*

  For example, to set limit on data directory on test-volume where data is a directory under the export directory:

```
# gluster volume quota test-volume limit-usage /data 10GB
Usage limit has been set on /data
```

Note

In a multi-level directory hierarchy, the minimum of disk limit in entire hierarchy will be considered for enforcement.

9.4. Displaying Disk Limit Information

You can display disk limit information on all the directories on which the limit is set.

To display disk limit information

- Display disk limit information of all the directories on which limit is set, using the following command:

  # gluster volume quota *VOLNAME* list

  For example, to see the set disks limit on test-volume:

```
# gluster volume quota test-volume list

 Path          Limit       Set Size
/Test/data    10 GB       6 GB
/Test/data1   10 GB       4 GB
```

NOTE that, the directory listed here is not absolute directory name, but relative path to the volume's root ('/'). For example, if 'test-volume' is mounted on '/mnt/glusterfs', then for the above example, '/Test/data' means, '/mnt/glusterfs/Test/data'

- Display disk limit information on a particular directory on which limit is set, using the following command:

  # gluster volume quota *VOLNAME* list */directory name*

  For example, to see the set limit on /data directory of test-volume:

```
# gluster volume quota test-volume list /data

Path           Limit      Set Size
/Test/data     10 GB       6 GB
```

9.5. Updating Memory Cache Size

For performance reasons, quota caches the directory sizes on client. You can set timeout indicating the maximum valid duration of directory sizes in cache, from the time they are populated.

For example: If there are multiple clients writing to a single directory, there are chances that some other client might write till the quota limit is exceeded. However, this new file-size may not get reflected in the client till size entry in cache has become stale because of timeout. If writes happen on this client during this duration, they are allowed even though they would lead to exceeding of quota-limits, since size in cache is not in sync with the actual size. When timeout happens, the size in cache is updated from servers and will be in sync and no further writes will be allowed. A timeout of zero will force fetching of directory sizes from server for every operation that modifies file data and will effectively disables directory size caching on client side.

To update the memory cache size

- Update the memory cache size using the following command:

  # gluster volume set *VOLNAME* features.quota-timeout *value*

  For example, to update the memory cache size for every 5 seconds on test-volume:

```
# gluster volume set test-volume features.quota-timeout 5
Set volume successful
```

9.6. Removing Disk Limit

You can remove set disk limit, if you do not want quota anymore.

To remove disk limit

- Remove disk limit set on a particular directory using the following command:

  # gluster volume quota *VOLNAME* remove */directory name*

  For example, to remove the disk limit on /data directory of test-volume:

```
# gluster volume quota test-volume remove /data
Usage limit set on /data is removed
```

Monitoring your GlusterFS Workload

You can monitor the GlusterFS volumes on different parameters. Monitoring volumes helps in capacity planning and performance tuning tasks of the GlusterFS volume. Using these information, you can identify and troubleshoot issues.

You can use Volume Top and Profile commands to view the performance and identify bottlenecks/ hotspots of each brick of a volume. This helps system administrators to get vital performance information whenever performance needs to be probed.

You can also perform statedump of the brick processes and nfs server process of a volume, and also view volume status and volume information.

10.1. Running GlusterFS Volume Profile Command

GlusterFS Volume Profile command provides an interface to get the per-brick I/O information for each File Operation (FOP) of a volume. The per brick information helps in identifying bottlenecks in the storage system.

This section describes how to run GlusterFS Volume Profile command by performing the following operations:

- *Section 10.1.1, “Start Profiling”*
- *Section 10.1.2, “Displaying the I/0 Information”*
- *Section 10.1.3, “Stop Profiling”*

10.1.1. Start Profiling

You must start the Profiling to view the File Operation information for each brick.

To start profiling:

- Start profiling using the following 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
```

When profiling on the volume is started, the following additional options are displayed in the Volume Info:

```
diagnostics.count-fop-hits: on

diagnostics.latency-measurement: on
```

10.1.2. Displaying the I/0 Information

You can view the I/O information of each brick.

To display I/O information:

- Display the I/O information using the following command:

 # gluster volume profile *VOLNAME* info

For example, to see the I/O information on test-volume:

```
# gluster volume profile test-volume info
Brick: Test:/export/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
```

10.1.3. Stop Profiling

You can stop profiling the volume, if you do not need profiling information anymore.

To stop profiling

- Stop profiling using 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
```

10.2. Running GlusterFS Volume TOP Command

GlusterFS Volume Top command allows you to view the glusterfs bricks' performance metrics like read, write, file open calls, file read calls, file write calls, directory open calls, and directory real calls. The top command displays up to 100 results.

This section describes how to run and view the results for the following GlusterFS Top commands:

- *Section 10.2.1, "Viewing Open fd Count and Maximum fd Count "*
- *Section 10.2.2, "Viewing Highest File Read Calls "*
- *Section 10.2.3, "Viewing Highest File Write Calls "*
- *Section 10.2.4, "Viewing Highest Open Calls on Directories "*
- *Section 10.2.5, "Viewing Highest Read Calls on Directory "*
- *Section 10.2.6, "Viewing List of Read Performance on each Brick "*
- *Section 10.2.7, "Viewing List of Write Performance on each Brick "*

10.2.1. Viewing Open fd Count and Maximum fd Count

You can view both current open fd count (list of files that are currently the most opened and the count) on the brick and the maximum open fd count (count of files that are the currently open and the count of 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 open fd metrics of all the bricks belonging to the volume will be displayed.

To view open fd count and maximum fd count:

- View open fd count and maximum fd count using the following command:

  ```
  # gluster volume top VOLNAME open [brick BRICK-NAME] [list-cnt cnt]
  ```

  For example, to view open fd count and maximum fd count on brick *server:/export* of *test-volume* and list top 10 open calls:

  ```
  # gluster volume top test-volume open brick server:/export/ list-cnt 10

  Brick: server:/export/dir1

  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
```

10.2.2. Viewing Highest File Read Calls

You can view highest read calls on each brick. If brick name is not specified, then by default, list of 100 files will be displayed.

To view highest file Read calls:

- View highest file Read calls using the following command:

  # gluster volume top *VOLNAME* read [brick *BRICK-NAME*] [list-cnt *cnt*]

  For example, to view 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
```

10.2.3. Viewing Highest File Write Calls

You can view list of files which has highest file write calls on each brick. If brick name is not specified, then by default, list of 100 files will be displayed.

To view highest file Write calls:

- View highest file Write calls using the following command:

  # gluster volume top *VOLNAME* write [brick *BRICK-NAME*] [list-cnt *cnt*]

  For example, to view 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
```

10.2.4. Viewing Highest Open Calls on Directories

You can view list of files which has highest open calls on directories of each brick. If brick name is not specified, then the metrics of all the bricks belonging to that volume will be displayed.

To view list of open calls on each directory

- View list of open calls on each directory using the following command:

  # gluster volume top *VOLNAME* opendir [brick *BRICK-NAME*] [list-cnt *cnt*]

  For example, to view 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
```

10.2.5. Viewing Highest Read Calls on Directory

You can view list of files which has highest directory read calls on each brick. If brick name is not specified, then the metrics of all the bricks belonging to that volume will be displayed.

To view list of highest directory read calls on each brick

- View list of highest directory read calls on each brick using the following command:

  # gluster volume top *VOLNAME* readdir [brick *BRICK-NAME*] [list-cnt *cnt*]

  For example, to view 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
```

10.2.6. Viewing List of Read Performance on each Brick

You can view the read throughput of files on each brick. 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 read throughput.

```
       ==========Read throughput file stats========

read         filename                          Time
through
put(MBp
s)

2570.00    /clients/client0/~dmtmp/PWRPNT/      -2011-01-31
           TRIDOTS.POT                       15:38:36.894610
2570.00    /clients/client0/~dmtmp/PWRPNT/      -2011-01-31
           PCBENCHM.PPT                      15:38:39.815310

2383.00    /clients/client2/~dmtmp/SEED/        -2011-01-31
           MEDIUM.FIL                        15:52:53.631499

2340.00    /clients/client0/~dmtmp/SEED/        -2011-01-31
           MEDIUM.FIL                        15:38:36.926198

2299.00   /clients/client0/~dmtmp/SEED/         -2011-01-31
          LARGE.FIL                          15:38:36.930445

2259.00   /clients/client0/~dmtmp/PARADOX/      -2011-01-31
          COURSES.X04                        15:38:40.549919

2221.00   /clients/client0/~dmtmp/PARADOX/      -2011-01-31
          STUDENTS.VAL                       15:52:53.298766

2221.00   /clients/client3/~dmtmp/SEED/         -2011-01-31
          COURSES.DB                         15:39:11.776780

2184.00   /clients/client3/~dmtmp/SEED/         -2011-01-31
          MEDIUM.FIL                         15:39:10.251764

2184.00   /clients/client5/~dmtmp/WORD/         -2011-01-31
          BASEMACH.DOC                       15:39:09.336572
```

This command will initiate a dd for the specified count and block size and measures the corresponding throughput.

To view list of read performance on each brick

- View list of read performance on each brick using the following command:

  # gluster volume top *VOLNAME* read-perf [bs *blk-size* count *count*] [brick *BRICK-NAME*] [list-cnt *cnt*]

  For example, to view read performance on brick server:/export/ of test-volume, 256 block size of count 1, and list count 10:

  # 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/    -2011-01-31
           TRIDOTS.POT                   15:38:36.896486

2570.00   /clients/client0/~dmtmp/PWRPNT/    -2011-01-31
           PCBENCHM.PPT                  15:38:39.815310

2383.00   /clients/client2/~dmtmp/SEED/      -2011-01-31
           MEDIUM.FIL                    15:52:53.631499

2340.00   /clients/client0/~dmtmp/SEED/      -2011-01-31
           MEDIUM.FIL                    15:38:36.926198

2299.00   /clients/client0/~dmtmp/SEED/      -2011-01-31
           LARGE.FIL                     15:38:36.930445

2259.00  /clients/client0/~dmtmp/PARADOX/    -2011-01-31
          COURSES.X04                    15:38:40.549919

2221.00  /clients/client9/~dmtmp/PARADOX/    -2011-01-31
          STUDENTS.VAL                   15:52:53.298766

2221.00  /clients/client8/~dmtmp/PARADOX/    -2011-01-31
         COURSES.DB                      15:39:11.776780

2184.00  /clients/client3/~dmtmp/SEED/       -2011-01-31
          MEDIUM.FIL                     15:39:10.251764

2184.00  /clients/client5/~dmtmp/WORD/       -2011-01-31
         BASEMACH.DOC                    15:39:09.336572
```

10.2.7. Viewing List of Write Performance on each Brick

You can view list of write throughput of files on each brick. 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 will initiate a dd for the specified count and block size and measures the corresponding throughput. To view list of write performance on each brick:

- View list of write performance on each brick using the following command:

  # gluster volume top *VOLNAME* write-perf [bs *blk-size* count *count*] [brick *BRICK-NAME*] [list-cnt *cnt*]

  For example, to view write performance on brick *server:/export/* of *test-volume*, 256 block size of count 1, and list count 10:

  # 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/     -2011-01-31
           SMALL.FIL                      15:39:09.171494

1008.00    /clients/client6/~dmtmp/SEED/     -2011-01-31
           LARGE.FIL                       15:39:09.73189

949.00    /clients/client0/~dmtmp/SEED/      -2011-01-31
          MEDIUM.FIL                      15:38:36.927426

936.00   /clients/client0/~dmtmp/SEED/       -2011-01-31
         LARGE.FIL                         15:38:36.933177
897.00   /clients/client5/~dmtmp/SEED/       -2011-01-31
         MEDIUM.FIL                        15:39:09.33628

897.00   /clients/client6/~dmtmp/SEED/       -2011-01-31
         MEDIUM.FIL                        15:39:09.27713

885.00   /clients/client0/~dmtmp/SEED/       -2011-01-31
          SMALL.FIL                      15:38:36.924271

528.00   /clients/client5/~dmtmp/SEED/       -2011-01-31
         LARGE.FIL                         15:39:09.81893

516.00   /clients/client6/~dmtmp/ACCESS/    -2011-01-31
         FASTENER.MDB                    15:39:01.797317
```

10.3. Displaying Volume Information

You can display information about a specific volume, or all volumes, as needed.

To display volume information

- Display information about a specific volume 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:/exp1
Brick2: server2:/exp2
Brick3: server3:/exp3
Brick4: server4:/exp4
```

- Display information about all volumes using the following command:

  `# gluster volume info all`

```
# gluster volume info all

Volume Name: test-volume
Type: Distribute
Status: Created
Number of Bricks: 4
Bricks:
Brick1: server1:/exp1
Brick2: server2:/exp2
```

```
Brick3: server3:/exp3
Brick4: server4:/exp4

Volume Name: mirror
Type: Distributed-Replicate
Status: Started
Number of Bricks: 2 X 2 = 4
Bricks:
Brick1: server1:/brick1
Brick2: server2:/brick2
Brick3: server3:/brick3
Brick4: server4:/brick4

Volume Name: Vol
Type: Distribute
Status: Started
Number of Bricks: 1
Bricks:
Brick: server:/brick6
```

10.4. Performing Statedump on a Volume

Statedump is a mechanism through which you can get details of all internal variables and state of the glusterfs process at the time of issuing the command.You can perform statedumps of the brick processes and nfs server process of a volume using the statedump command. The following options can be used to determine what information is to be dumped:

- **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 fd tables of the volume.
- **inode** - Dumps the inode tables of the volume.

To display volume statedump

- Display statedump of a volume or NFS server using the following command:

  # gluster volume statedump *VOLNAME* [nfs] [all|mem|iobuf|callpool|priv|fd|inode]

  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 **server.statedump-path** volume option. The naming convention of the dump file is **<brick-path>.<brick-pid>.dump**.

- By defult, the output of the statedump is stored at **/tmp/<brickname.PID.dump>** file on that particular server. Change the directory of the statedump file using the following command:

  # gluster volume set *VOLNAME* server.statedump-path *path*

For example, to change the location of the statedump file of test-volume:

```
# gluster volume set test-volume server.statedump-path /usr/local/var/log/glusterfs/dumps/
Set volume successful
```

You can view the changed path of the statedump file using the following command:

 # gluster volume info *VOLNAME*

10.5. 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, 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 following 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 fd (file descriptors) tables of the volume.
- **callpool** - Displays the pending calls of the volume.

To display volume status

- Display information about a specific volume using the following command:

  # gluster volume status [all|*VOLNAME* [*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
BRICK                           PORT   ONLINE   PID
------------------------------------------------------------
arch:/export/1                  24009   Y       22445
------------------------------------------------------------
arch:/export/2                  24010   Y       22450
```

- Display information about all volumes using the following command:

  # gluster volume status all

```
# gluster volume status all
STATUS OF VOLUME: volume-test
BRICK                           PORT   ONLINE   PID
------------------------------------------------------------
arch:/export/4                  24010   Y       22455

STATUS OF VOLUME: test-volume
BRICK                           PORT   ONLINE   PID
------------------------------------------------------------
```

```
arch:/export/1                24009   Y       22445
------------------------------------------------------------
arch:/export/2                24010   Y       22450
```

- Display additional information about the bricks using the following command:

  # gluster volume status *VOLNAME* detail

  For example, to display additional information about the bricks of test-volume:

```
# gluster volume status test-volume details
STATUS OF VOLUME: test-volume
------------------------------------------
Brick                : arch:/export/1
Port                 : 24009
Online               : Y
Pid                  : 16977
File System          : rootfs
Device               : rootfs
Mount Options        : rw
Disk Space Free      : 13.8GB
Total Disk Space     : 46.5GB
Inode Size           : N/A
Inode Count          : N/A
Free Inodes          : N/A

Number of Bricks: 1
Bricks:
Brick: server:/brick6
```

- Display the list of clients accessing the volumes using the following 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 : arch:/export/1
Clients connected : 2
Hostname          Bytes Read   BytesWritten
--------          ---------    ------------
127.0.0.1:1013    776          676
127.0.0.1:1012    50440        51200
```

- Display the memory usage and memory pool details of the bricks using the following command:

  # gluster volume status *VOLNAME* mem

  For example, to display the memory usage and memory pool details of the bricks of test-volume:

```
Memory status for volume : test-volume
----------------------------------------------
Brick : arch:/export/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 following command:

  # gluster volume status *VOLNAME* inode

  For example, to display the inode tables of the test-volume:

```
# gluster volume status test-volume inode
inode tables for volume test-volume
----------------------------------------------
Brick : arch:/export/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 fd tables of the volume using the following command:

  # gluster volume status *VOLNAME* fd

  For example, to display the open fd tables of the test-volume:

```
# gluster volume status test-volume fd

FD tables for volume test-volume
----------------------------------------------
Brick : arch:/export/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
```

- Display the pending calls of the volume using the following command:

  # gluster volume status *VOLNAME* callpool

  Each call has a call stack containing call frames.

  For example, to display the pending calls of test-volume:

```
# gluster volume status test-volume

Pending calls for volume test-volume
----------------------------------------------
Brick : arch:/export/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
```

POSIX Access Control Lists

POSIX Access Control Lists (ACLs) allows you to assign different permissions for different users or groups even though they do not correspond to the original owner or the owning group.

For example: User john creates a file but does not want to allow anyone to do anything with this file, except another user, antony (even though there are other users that belong to the group john).

This means, in addition to the file owner, the file group, and others, additional users and groups can be granted or denied access by using POSIX ACLs.

11.1. Activating POSIX ACLs Support

To use POSIX ACLs for a file or directory, the mount point where the file or directory exists, must be mounted with POSIX ACLs support.

11.1.1. Activating POSIX ACLs Support on Sever

To mount the backend export directories for POSIX ACLs support, use the following command:

```
# mount -o acl device-namepartition
```

If the backend export directory is already mounted, use the following command:

```
# mount -oremount,acl device-namepartition
```

For example:

```
# mount -o acl /dev/sda1 /export1
```

Alternatively, if the partition is listed in the /etc/fstab file, add the following entry for the partition to include the POSIX ACLs option:

```
LABEL=/work /export1 xfs rw,acl 1 4
```

11.1.2. Activating POSIX ACLs Support on Client

To mount the glusterfs volume with POSIX ACLs support, use the following command:

```
# mount –t glusterfs -o acl severname:/volume-idmount point
```

For example:

```
# mount -t glusterfs -o acl 198.192.198.234:/glustervolume /mnt/gluster
```

11.2. Setting POSIX ACLs

You can set two types of POSIX ACLs, that is, access ACLs and default ACLs. You can use access ACLs to grant permission for a specific file or directory. You can use default ACLs only on a directory but if a file inside that directory does not have an ACLs, it inherits the permissions of the default ACLs of the directory.

You can set ACLs for per user, per group, for users not in the user group for the file, and via the effective right mask.

11.2.1. Setting Access ACLs

You can apply access ACLs to grant permission for both files and directories.

To set or modify Access ACLs

You can set or modify access ACLs use the following command:

```
# setfacl –m entry type file
```

The ACL entry types are the POSIX ACLs representations of owner, group, and other.

Permissions must be a combination of the characters **r** (read), **w** (write), and **x** (execute). You must specify the ACL entry in the following format and can specify multiple entry types separated by commas.

ACL Entry	Description
u:uid:<permission>	Sets the access ACLs for a user. You can specify user name or UID
g:gid:<permission>	Sets the access ACLs for a group. You can specify group name or GID.
m:<permission>	Sets the effective rights mask. The mask is the combination of all access permissions of the owning group and all of the user and group entries.
o:<permission>	Sets the access ACLs for users other than the ones in the group for the file.

If a file or directory already has an POSIX ACLs, and the setfacl command is used, the additional permissions are added to the existing POSIX ACLs or the existing rule is modified.

For example, to give read and write permissions to user antony:

```
# setfacl -m u:antony:rw /mnt/gluster/data/testfile
```

11.2.2. Setting Default ACLs

You can apply default ACLs only to directories. They determine the permissions of a file system objects that inherits from its parent directory when it is created.

To set default ACLs

You can set default ACLs for files and directories using the following command:

```
# setfacl –m –-set entry type directory
```

For example, to set the default ACLs for the /data directory to read for users not in the user group:

```
# setfacl –m --set o::r /mnt/gluster/data
```

Note

An access ACLs set for an individual file can override the default ACLs permissions.

Effects of a Default ACLs

The following are the ways in which the permissions of a directory's default ACLs are passed to the files and subdirectories in it:

- A subdirectory inherits the default ACLs of the parent directory both as its default ACLs and as an access ACLs.
- A file inherits the default ACLs as its access ACLs.

11.3. Retrieving POSIX ACLs

You can view the existing POSIX ACLs for a file or directory.

To view existing POSIX ACLs

- View the existing access ACLs of a file using the following command:

  # getfacl *path/filename*

  For example, to view the existing POSIX ACLs for sample.jpg

```
# getfacl /mnt/gluster/data/test/sample.jpg
# owner: antony
# group: antony
user::rw-
group::rw-
other::r--
```

- View the default ACLs of a directory using the following command:

  # getfacl *directory name*

  For example, to view the existing ACLs for /data/doc

```
# 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
```

11.4. Removing POSIX ACLs

To remove all the permissions for a user, groups, or others, use the following command:

 # setfacl -x *ACL entry type file*

For example, to remove all permissions from the user antony:

 # setfacl -x u:antony /mnt/gluster/data/test-file

11.5. Samba and ACLs

If you are using Samba to access GlusterFS FUSE mount, then POSIX ACLs are enabled by default. Samba has been compiled with the **`--with-acl-support`** option, so no special flags are required when accessing or mounting a Samba share.

11.6. NFS and ACLs

Currently we do not support ACLs configuration through NFS, i.e. setfacl and getfacl commands do not work. However, ACLs permissions set using Gluster Native Client applies on NFS mounts.

Managing Unified File and Object Storage

Unified File and Object Storage (UFO) unifies NAS and object storage technology. It 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.

Unified File and Object Storage is built upon Openstack's Object Storage Swift. Open Stack Object Storage allows users to store and retrieve files and content through a simple Web Service (REST: Representational State Transfer) interface as objects and GlusterFS, allows users to store and retrieve files using Native Fuse and NFS mounts. It uses GlusterFS as a backend file system for Open Stack Swift. It also leverages on Open Stack Swift's web interface for storing and retrieving files over the web combined with GlusterFS features like scalability and high availability, replication, elastic volume management for data management at disk level.

Unified File and Object Storage 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 including NFS and CIFS. 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 GlusterFS Unified File and Object Storage 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 Object Storage is scalable object storage system and it is not a traditional file system. You will not be able to mount this system like traditional SAN or NAS volumes and perform POSIX compliant operations.

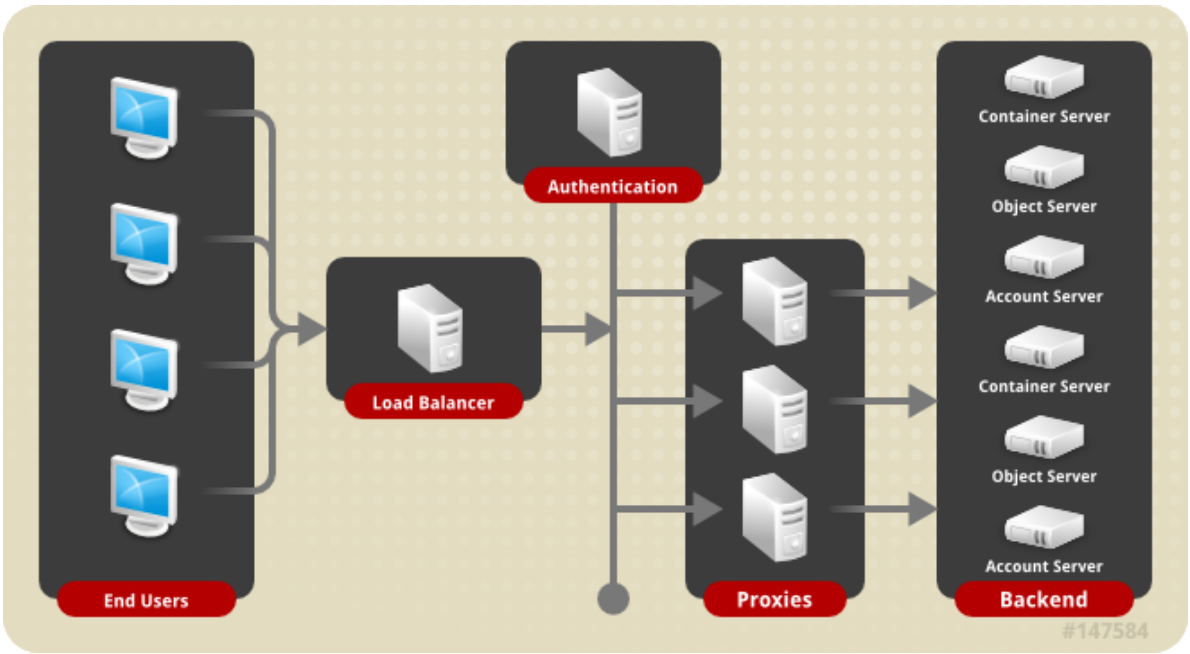


Figure 12.1. Unified File and Object Storage Architecture

12.1. Components of Object Storage

The major components of Object Storage are:

Proxy Server

All REST requests to the UFO are routed through the Proxy Server.

Objects and Containers

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).

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. Data must be stored in a container and hence objects are created within a container.

It implements objects as files and directories under the container. The object name is a '/' separated path and UFO maps it to directories until the last name in the path, which is marked as a file. With this approach, objects can be accessed as files and directories from native GlusterFS (FUSE) or NFS mounts by providing the '/' separated path.

Accounts and Account Servers

The OpenStack Object Storage system is designed to be used by many different storage consumers. Each user is associated with one or more accounts and must identify themselves using an authentication system. While authenticating, users must provide the name of the account for which the authentication is requested.

UFO implements accounts as GlusterFS volumes. So, when a user is granted read/write permission on an account, it means that that user has access to all the data available on that GlusterFS volume.

Authentication and Access Permissions

You must authenticate against an authentication service to receive OpenStack Object Storage connection parameters and an authentication token. The token must be passed in for all subsequent container or object operations. One authentication service that you can use as a middleware example is called **`tempauth`**.

By default, each user has their own storage account and has full access to that account. Users must authenticate with their credentials as described above, but once authenticated they can manage containers and objects within that account. If a user wants to access the content from another account, they must have API access key or a session token provided by their authentication system.

12.2. Advantages of using GlusterFS Unified File and Object Storage

The following are the advantages of using GlusterFS UFO:

- No limit on upload and download files sizes as compared to Open Stack Swift which limits the object size to 5GB.
- A unified view of data across NAS and Object Storage technologies.
- Using GlusterFS's UFO has other advantages like the following:
  - High availability
  - Scalability
  - Replication
  - Elastic Volume management

12.3. Preparing to Deploy Unified File and Object Storage

This section provides information on pre-requisites and list of dependencies that will be installed during the installation of Unified File and Object Storage.

12.3.1. Pre-requisites

GlusterFS's Unified File and Object Storage needs **user_xattr** support from the underlying disk file system. Use the following command to enable **user_xattr** for GlusterFS bricks backend:

```
# mount –o remount,user_xattr device name
```

For example,

```
# mount –o remount,user_xattr /dev/hda1
```

12.3.2. Dependencies

The following packages are installed on GlusterFS when you install Unified File and Object Storage:

- curl
- memcached
- openssl
- xfsprogs
- python2.6
- pyxattr
- python-configobj
- python-setuptools
- python-simplejson
- python-webob
- python-eventlet
- python-greenlet
- python-pastedeploy
- python-netifaces

12.4. Installing and Configuring Unified File and Object Storage

This section provides instructions on how to install and configure Unified File and Object Storage in your storage environment.

12.4.1. Installing Unified File and Object Storage

To install Unified File and Object Storage:

1. Download **rhel_install.sh** install script from *http://download.gluster.com/pub/gluster/glusterfs/3.2/UFO/* .

2. Run **rhel_install.sh** script using the following command:

   ```
   # sh rhel_install.sh
   ```

3. Download **swift-1.4.5-1.noarch.rpm** and **swift-plugin-1.0.-1.el6.noarch.rpm** files from *http://download.gluster.com/pub/gluster/glusterfs/3.2/UFO/*.

4. Install **swift-1.4.5-1.noarch.rpm** and **swift-plugin-1.0.-1.el6.noarch.rpm** using the following commands:

   ```
   # rpm -ivh swift-1.4.5-1.noarch.rpm
   ```

   ```
   # rpm -ivh swift-plugin-1.0.-1.el6.noarch.rpm
   ```

Note

You must repeat the above steps on all the machines on which you want to install Unified File and Object Storage. If you install the Unified File and Object Storage on multiple servers, you can use a load balancer like pound, nginx, and so on to distribute the request across the machines.

12.4.2. Adding Users

The authentication system allows the administrator to grant different levels of access to different users based on the requirement. The following are the types of user permissions:

- admin user
- normal user

Admin user has read and write permissions on the account. By default, a normal user has no read or write permissions. A normal user can only authenticate itself to get a Auth-Token. Read or write permission are provided through ACLs by the admin users.

Add a new user by adding the following entry in **/etc/swift/proxy-server.conf** file:

```
user_<account-name>_<user-name> = <password> [.admin]
```

For example,

```
user_test_tester = testing .admin
```

Note

During installation, the installation script adds few sample users to the **proxy-server.conf** file. It is highly recommended that you remove all the default sample user entries from the configuration file.

For more information on setting ACLs, see *Section 12.5.3.6, " Setting ACLs on Container "*.

12.4.3. Configuring 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 route 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 configurable options pertaining to proxy server are stored in **/etc/swift/proxy-server.conf**. The following is the sample **proxy-server.conf** file:

```
[app:proxy-server]
use = egg:swift#proxy
allow_account_management=true
account_autocreate=true

[filter:tempauth]
use = egg:swift#tempauth
 user_admin_admin=admin.admin.reseller_admin
user_test_tester=testing.admin
user_test2_tester2=testing2.admin
user_test_tester3=testing3

[filter:healthcheck]
use = egg:swift#healthcheck

[filter:cache]
use = egg:swift#memcache
```

By default, GlusterFS's Unified File and Object Storage is configured to support HTTP protocol and uses temporary authentication to authenticate the HTTP requests.

12.4.4. Configuring Authentication System

Proxy server must be configured to authenticate using **tempauth** .

12.4.5. Configuring Proxy Server for HTTPS

By default, proxy server only handles HTTP request. 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
```

3. Restart the servers using the following commands:

```
swift-init main stop
swift-init main start
```

The following are the configurable options:

Table 12.1. proxy-server.conf Default Options in the [DEFAULT] section

Option	Default	Description
bind_ip	0.0.0.0	IP Address for server to bind
bind_port	80	Port for server to bind
swift_dir	/etc/swift	Swift configuration directory
workers	1	Number of workers to fork
user	swift	swift user
cert_file		Path to the ssl .crt
key_file		Path to the ssl .key

Table 12.2. proxy-server.conf Server Options in the [proxy-server] section

Option	Default	Description
use		paste.deploy entry point for the container server. For most cases, this should be **`egg:swift#container`**.
log_name	proxy-server	Label used when logging
log_facility	LOG_LOCAL0	Syslog log facility
log_level	INFO	Log level
log_headers	True	If True, log headers in each request
recheck_account_existence	60	Cache timeout in seconds to send memcached for account existence
recheck_container_existence	60	Cache timeout in seconds to send memcached for container existence
object_chunk_size	65536	Chunk size to read from object servers
client_chunk_size	65536	Chunk size to read from clients
memcache_servers	127.0.0.1:11211	Comma separated list of memcached servers ip:port
node_timeout	10	Request timeout to external services

Option	Default	Description
client_timeout	60	Timeout to read one chunk from a client
conn_timeout	0.5	Connection timeout to external services
error_suppression_interval	60	Time in seconds that must elapse since the last error for a node to be considered no longer error limited
error_suppression_limit	10	Error count to consider a node error limited
allow_account_management	false	Whether account **PUT**s and **DELETE**s are even callable

12.4.6. Configuring Object Server

The Object Server is a very simple blob storage server that can store, retrieve, and delete objects stored on local devices. Objects are stored as binary files on the file system with metadata stored in the file's extended attributes (xattrs). This requires that the underlying file system choice for object servers support xattrs on files.

The configurable options pertaining Object Server are stored in the file **`/etc/swift/object-server/1.conf`**. The following is the sample **`object-server/1.conf`** file:

```
[DEFAULT]
devices = /srv/1/node
mount_check = false
bind_port = 6010
user = root
log_facility = LOG_LOCAL2

[pipeline:main]
pipeline = gluster object-server

[app:object-server]
use = egg:swift#object

[filter:gluster]
use = egg:swift#gluster

[object-replicator]
vm_test_mode = yes

[object-updater]
[object-auditor]
```

The following are the configurable options:

Table 12.3. object-server.conf Default Options in the [DEFAULT] section

Option	Default	Description
swift_dir	/etc/swift	Swift configuration directory
devices	/srv/node	Mount parent directory where devices are mounted
mount_check	true	Whether or not check if the devices are mounted to prevent

Option	Default	Description
		accidentally writing to the root device
bind_ip	0.0.0.0	IP Address for server to bind
bind_port	6000	Port for server to bind
workers	1	Number of workers to fork

Table 12.4. object-server.conf Server Options in the [object-server] section

Option	Default	Description
use		paste.deploy entry point for the object server. For most cases, this should be **`egg:swift#object`**.
log_name	object-server	log name used when logging
log_facility	LOG_LOCAL0	Syslog log facility
log_level	INFO	Logging level
log_requests	True	Whether or not to log each request
user	swift	swift user
node_timeout	3	Request timeout to external services
conn_timeout	0.5	Connection timeout to external services
network_chunk_size	65536	Size of chunks to read or write over the network
disk_chunk_size	65536	Size of chunks to read or write to disk
max_upload_time	65536	Maximum time allowed to upload an object
slow	0	If > 0, Minimum time in seconds for a **PUT** or **DELETE** request to complete

12.4.7. Configuring Container Server

The Container Server's primary job is to handle listings of objects. The listing is done by querying the GlusterFS mount point with path. This query returns a list of all files and directories present under that container.

The configurable options pertaining to container server are stored in **`/etc/swift/container-server/1.conf`** file. The following is the sample **`container-server/1.conf`** file:

```
[DEFAULT]
devices = /srv/1/node
mount_check = false
bind_port = 6011
user = root
log_facility = LOG_LOCAL2

[pipeline:main]
```

```
pipeline = gluster container-server

[app:container-server]
use = egg:swift#container

[filter:gluster]
use = egg:swift#gluster

[container-replicator]
[container-updater]
[container-auditor]
```

The following are the configurable options:

Table 12.5. container-server.conf Default Options in the [DEFAULT] section

Option	Default	Description
swift_dir	/etc/swift	Swift configuration directory
devices	/srv/node	Mount parent directory where devices are mounted
mount_check	true	Whether or not check if the devices are mounted to prevent accidentally writing to the root device
bind_ip	0.0.0.0	IP Address for server to bind
bind_port	6001	Port for server to bind
workers	1	Number of workers to fork
user	swift	Swift user

Table 12.6. container-server.conf Server Options in the [container-server] section

Option	Default	Description
use		paste.deploy entry point for the container server. For most cases, this should be **`egg:swift#container`**.
log_name	container-server	Label used when logging
log_facility	LOG_LOCAL0	Syslog log facility
log_level	INFO	Logging level
node_timeout	3	Request timeout to external services
conn_timeout	0.5	Connection timeout to external services

12.4.8. Configuring Account Server

The Account Server is very similar to the Container Server, except that it is responsible for listing of containers rather than objects. In UFO, each gluster volume is an account.

The configurable options pertaining to account server are stored in **`/etc/swift/account-server/1.conf`** file. The following is the sample **`account-server/1.conf`** file:

```
[DEFAULT]
devices = /srv/1/node
```

```
mount_check = false
bind_port = 6012
user = root
log_facility = LOG_LOCAL2

[pipeline:main]
pipeline = gluster account-server

[app:account-server]
use = egg:swift#account

[filter:gluster]
use = egg:swift#gluster

[account-replicator]
vm_test_mode = yes

[account-auditor]
[account-reaper]
```

The following are the configurable options:

Table 12.7. account-server.conf Default Options in the [DEFAULT] section

Option	Default	Description
swift_dir	/etc/swift	Swift configuration directory
devices	/srv/node	mount parent directory where devices are mounted
mount_check	true	Whether or not check if the devices are mounted to prevent accidentally writing to the root device
bind_ip	0.0.0.0	IP Address for server to bind
bind_port	6002	Port for server to bind
workers	1	Number of workers to fork
user	swift	Swift user

Table 12.8. account-server.conf Server Options in the [account-server] section

Option	Default	Description
use		paste.deploy entry point for the container server. For most cases, this should be **`egg:swift#container`**.
log_name	account-server	Label used when logging
log_facility	LOG_LOCAL0	Syslog log facility
log_level	INFO	Logging level

12.4.9. Starting and Stopping Server

You must start the server manually when system reboots and whenever you update/modify the configuration files.

- To start the server, enter the following command:

```
# swift_init main start
```

- To stop the server, enter the following command:

```
# swift_init main stop
```

12.5. Working with Unified File and Object Storage

This section describes the REST API for administering and managing Object Storage. All requests will be directed to the host and URL described in the **X-Storage-URL HTTP** header obtained during successful authentication.

12.5.1. Configuring Authenticated Access

Authentication is the process of proving identity to the system. To use the REST interface, you must obtain an authorization token using GET method and supply it with v1.0 as the path.

Each REST request against the Object Storage system requires the addition of a specific authorization token HTTP x-header, defined as X-Auth-Token. The storage URL and authentication token are returned in the headers of the response.

- To authenticate, run the following command:

```
GET auth/v1.0 HTTP/1.1
Host: <auth URL>
X-Auth-User: <account name>:<user name>
X-Auth-Key: <user-Password>
```

For example,

```
GET auth/v1.0 HTTP/1.1
Host: auth.example.com
X-Auth-User: test:tester
X-Auth-Key: testing

HTTP/1.1 200 OK
X-Storage-Url: https:/example.storage.com:443/v1/AUTH_test
X-Storage-Token: AUTH_tkde3ad38b087b49bbbac0494f7600a554
X-Auth-Token: AUTH_tkde3ad38b087b49bbbac0494f7600a554
Content-Length: 0
Date: Wed, 10 jul 2011 06:11:51 GMT
```

To authenticate access using cURL (for the above example), run the following command:

```
curl -v -H 'X-Storage-User: test:tester' -H 'X-Storage-Pass:testing' -k
https://auth.example.com:443/auth/v1.0
```

The X-Auth-Url has to be parsed and used in the connection and request line of all subsequent requests to the server. In the example output, users connecting to server will send most container/object requests with a host header of example.storage.com and the request line's version and account as v1/AUTH_test.

Note

The authentication tokens are valid for a 24 hour period.

12.5.2. Working with Accounts

This section describes the list of operations you can perform at the account level of the URL.

12.5.2.1. Displaying Container Information

You can list the objects of a specific container, or all containers, as needed using GET command. You can use the following optional parameters with GET request to refine the results:

Parameter	Description
limit	Limits the number of results to at most *n* value.
marker	Returns object names greater in value than the specified marker.
format	Specify either json or xml to return the respective serialized response.

To display container information

- List all the containers of an account using the following command:

```
GET /<apiversion>/<account> HTTP/1.1
Host: <storage URL>
X-Auth-Token: <authentication-token-key>
```

For example,

```
GET /v1/AUTH_test HTTP/1.1
Host: example.storage.com
X-Auth-Token: AUTH_tkd3ad38b087b49bbbac0494f7600a554

HTTP/1.1 200 Ok
Date: Wed, 13 Jul 2011 16:32:21 GMT
Server: Apache
Content-Type: text/plain; charset=UTF-8
Content-Length: 39

songs
movies
documents
reports
```

To display container information using cURL (for the above example), run the following command:

```
curl -v -X GET -H 'X-Auth-Token: AUTH_tkde3ad38b087b49bbbac0494f7600a554'
https://example.storage.com:443/v1/AUTH_test -k
```

12.5.2.2. Displaying Account Metadata Information

You can issue HEAD command to the storage service to view the number of containers and the total bytes stored in the account.

- To display containers and storage used, run the following command:

```
HEAD /<apiversion>/<account> HTTP/1.1
Host: <storage URL>
X-Auth-Token: <authentication-token-key>
```

For example,

```
HEAD /v1/AUTH_test HTTP/1.1
Host: example.storage.com
X-Auth-Token: AUTH_tkd3ad38b087b49bbbac0494f7600a554

HTTP/1.1 204 No Content
Date: Wed, 13 Jul 2011 16:52:21 GMT
Server: Apache
X-Account-Container-Count: 4
X-Account-Total-Bytes-Used: 394792
```

To display account metadata information using cURL (for the above example), run the following command:

```
curl -v -X HEAD -H 'X-Auth-Token:
AUTH_tkde3ad38b087b49bbbac0494f7600a554'
https://example.storage.com:443/v1/AUTH_test -k
```

12.5.3. Working with Containers

This section describes the list of operations you can perform at the container level of the URL.

12.5.3.1. Creating Containers

You can use PUT command to create containers. Containers are the storage folders for your data. The URL encoded name must be less than 256 bytes and cannot contain a forward slash '/' character.

- To create a container, run the following command:

```
PUT /<apiversion>/<account>/<container>/ HTTP/1.1
Host: <storage URL>
X-Auth-Token: <authentication-token-key>
```

For example,

```
PUT /v1/AUTH_test/pictures/ HTTP/1.1
Host: example.storage.com
X-Auth-Token: AUTH_tkd3ad38b087b49bbbac0494f7600a554
HTTP/1.1 201 Created

Date: Wed, 13 Jul 2011 17:32:21 GMT
Server: Apache
Content-Type: text/plain; charset=UTF-8
```

To create container using cURL (for the above example), run the following command:

```
curl -v -X PUT -H 'X-Auth-Token:
AUTH_tkde3ad38b087b49bbbac0494f7600a554'
https://example.storage.com:443/v1/AUTH_test/pictures -k
```

The status code of 201 (Created) indicates that you have successfully created the container. If a container with same is already existed, the status code of 202 is displayed.

12.5.3.2. Displaying Objects of a Container

You can list the objects of a container using GET command. You can use the following optional parameters with GET request to refine the results:

Parameter	Description
limit	Limits the number of results to at most *n* value.
marker	Returns object names greater in value than the specified marker.
prefix	Displays the results limited to object names beginning with the substring x. beginning with the substring x.
path	Returns the object names nested in the pseudo path.
format	Specify either json or xml to return the respective serialized response.
delimiter	Returns all the object names nested in the container.

To display objects of a container

- List objects of a specific container using the following command:

```
GET /<apiversion>/<account>/<container>[parm=value] HTTP/1.1
Host: <storage URL>
X-Auth-Token: <authentication-token-key>
```

For example,

```
GET /v1/AUTH_test/images HTTP/1.1
Host: example.storage.com
X-Auth-Token: AUTH_tkd3ad38b087b49bbbac0494f7600a554

HTTP/1.1 200 Ok
Date: Wed, 13 Jul 2011 15:42:21 GMT
Server: Apache
Content-Type: text/plain; charset=UTF-8
Content-Length: 139

sample file.jpg
test-file.pdf
You and Me.pdf
Puddle of Mudd.mp3
Test Reports.doc
```

To display objects of a container using cURL (for the above example), run the following command:

```
curl -v -X GET-H 'X-Auth-Token: AUTH_tkde3ad38b087b49bbbac0494f7600a554'
https://example.storage.com:443/v1/AUTH_test/images -k
```

12.5.3.3. Displaying Container Metadata Information

You can issue HEAD command to the storage service to view the number of objects in a container and the total bytes of all the objects stored in the container.

- To display list of objects and storage used, run the following command:

```
HEAD /<apiversion>/<account>/<container> HTTP/1.1
Host: <storage URL>
X-Auth-Token: <authentication-token-key>
```

For example,

```
HEAD /v1/AUTH_test/images HTTP/1.1
Host: example.storage.com
X-Auth-Token: AUTH_tkd3ad38b087b49bbbac0494f7600a554

HTTP/1.1 204 No Content
Date: Wed, 13 Jul 2011 19:52:21 GMT
Server: Apache
X-Account-Object-Count: 8
X-Container-Bytes-Used: 472
```

To display list of objects and storage used in a container using cURL (for the above example), run the following command:

```
curl -v -X HEAD -H 'X-Auth-Token:
AUTH_tkde3ad38b087b49bbbac0494f7600a554'
https://example.storage.com:443/v1/AUTH_test/images -k
```

12.5.3.4. Deleting Container

You can use DELETE command to permanently delete containers. The container must be empty before it can be deleted.

You can issue HEAD command to determine if it contains any objects.

- To delete a container, run the following command:

```
DELETE /<apiversion>/<account>/<container>/ HTTP/1.1
Host: <storage URL>
X-Auth-Token: <authentication-token-key>
```

For example,

```
DELETE /v1/AUTH_test/pictures HTTP/1.1
Host: example.storage.com
X-Auth-Token: AUTH_tkd3ad38b087b49bbbac0494f7600a554

HTTP/1.1 204 No Content
Date: Wed, 13 Jul 2011 17:52:21 GMT
Server: Apache
Content-Length: 0
Content-Type: text/plain; charset=UTF-8
```

To delete a container using cURL (for the above example), run the following command:

```
curl -v -X DELETE -H 'X-Auth-Token:
AUTH_tkde3ad38b087b49bbbac0494f7600a554'
https://example.storage.com:443/v1/AUTH_test/pictures -k
```

The status code of 204 (No Content) indicates that you have successfully deleted the container. If that container does not exist, the status code 404 (Not Found) is displayed, and if the container is not empty, the status code 409 (Conflict) is displayed.

12.5.3.5. Updating Container Metadata

You can update the metadata of container using POST operation, metadata keys should be prefixed with 'x-container-meta'.

- To update the metadata of the object, run the following command:

```
POST /<apiversion>/<account>/<container> HTTP/1.1
Host: <storage URL>
X-Auth-Token: <Authentication-token-key>
X-Container-Meta-<key>: <new value>
X-Container-Meta-<key>: <new value>
```

For example,

```
POST /v1/AUTH_test/images HTTP/1.1
Host: example.storage.com
X-Auth-Token: AUTH_tkd3ad38b087b49bbbac0494f7600a554
X-Container-Meta-Zoo: Lion
X-Container-Meta-Home: Dog

HTTP/1.1 204 No Content
Date: Wed, 13 Jul 2011 20:52:21 GMT
Server: Apache
Content-Type: text/plain; charset=UTF-8
```

To update the metadata of the object using cURL (for the above example), run the following command:

```
curl -v -X POST -H 'X-Auth-Token:
AUTH_tkde3ad38b087b49bbbac0494f7600a554'
https://example.storage.com:443/v1/AUTH_test/images -H ' X-Container-Meta-Zoo: Lion' -H
 'X-Container-Meta-Home: Dog' -k
```

The status code of 204 (No Content) indicates the container's metadata is updated successfully. If that object does not exist, the status code 404 (Not Found) is displayed.

12.5.3.6. Setting ACLs on Container

You can set the container access control list by using POST command on container with **`x-container-read`** and **`x-container-write`** keys.

The ACL format is **`[item[,item...]]`**. Each item can be a group name to give access to or a referrer designation to grant or deny based on the HTTP Referer header.

The referrer designation format is: **`.r:[-]value`**.

The .r can also be **`.ref`**, **`.referer`**, or .**`referrer`**; though it will be shortened to.r for decreased character count usage. The value can be **`*`** to specify any referrer host is allowed access. The leading minus sign (-) indicates referrer hosts that should be denied access.

Examples of valid ACLs:

```
.r:*
```

```
.r:*,bobs_account,sues_account:sue
bobs_account,sues_account:sue
```

Examples of invalid ACLs:

```
.r:
.r:-
```

By default, allowing read access via **.r** will not allow listing objects in the container but allows retrieving objects from the container. To turn on listings, use the .**rlistings** directive. Also, **.r** designations are not allowed in headers whose names include the word write.

For example, to set all the objects access rights to "public# inside the container using cURL (for the above example), run the following command:

```
curl -v -X POST -H 'X-Auth-Token:
AUTH_tkde3ad38b087b49bbbac0494f7600a554'
https://example.storage.com:443/v1/AUTH_test/images
-H 'X-Container-Read: .r:*' -k
```

12.5.4. Working with Objects

An object represents the data and any metadata for the files stored in the system. Through the REST interface, metadata for an object can be included by adding custom HTTP headers to the request and the data payload as the request body. Objects name should not exceed 1024 bytes after URL encoding.

This section describes the list of operations you can perform at the object level of the URL.

12.5.4.1. Creating or Updating Object

You can use PUT command to write or update an object's content and metadata.

You can verify the data integrity by including an MD5checksum for the object's data in the ETag header. ETag header is optional and can be used to ensure that the object's contents are stored successfully in the storage system.

You can assign custom metadata to objects by including additional HTTP headers on the PUT request. The objects created with custom metadata via HTTP headers are identified with the**X-Object- Meta**- prefix.

- To create or update an object, run the following command:

```
PUT /<apiversion>/<account>/<container>/<object> HTTP/1.1
Host: <storage URL>
X-Auth-Token: <authentication-token-key>
ETag: da1e100dc9e7becc810986e37875ae38
Content-Length: 342909
X-Object-Meta-PIN: 2343
```

For example,

```
PUT /v1/AUTH_test/pictures/dog HTTP/1.1
Host: example.storage.com
X-Auth-Token: AUTH_tkd3ad38b087b49bbbac0494f7600a554
ETag: da1e100dc9e7becc810986e37875ae38

HTTP/1.1 201 Created
```

```
Date: Wed, 13 Jul 2011 18:32:21 GMT
Server: Apache
ETag: da1e100dc9e7becc810986e37875ae38
Content-Length: 0
Content-Type: text/plain; charset=UTF-8
```

To create or update an object using cURL (for the above example), run the following command:

```
curl -v -X PUT -H 'X-Auth-Token:
AUTH_tkde3ad38b087b49bbbac0494f7600a554'
https://example.storage.com:443/v1/AUTH_test/pictures/dog -H 'Content-
Length: 0' -k
```

The status code of 201 (Created) indicates that you have successfully created or updated the object. If there is a missing content-Length or Content-Type header in the request, the status code of 412 (Length Required) is displayed. (Optionally) If the MD5 checksum of the data written to the storage system does not match the ETag value, the status code of 422 (Unprocessable Entity) is displayed.

12.5.4.1.1. Chunked Transfer Encoding

You can upload data without knowing the size of the data to be uploaded. You can do this by specifying an HTTP header of Transfer-Encoding: chunked and without using a Content-Length header.

You can use this feature while doing a DB dump, piping the output through gzip, and then piping the data directly into Object Storage without having to buffer the data to disk to compute the file size.

- To create or update an object, run the following command:

```
PUT /<apiversion>/<account>/<container>/<object> HTTP/1.1
Host: <storage URL>
X-Auth-Token: <authentication-token-key>
Transfer-Encoding: chunked
X-Object-Meta-PIN: 2343
```

For example,

```
PUT /v1/AUTH_test/pictures/cat HTTP/1.1
Host: example.storage.com
X-Auth-Token: AUTH_tkd3ad38b087b49bbbac0494f7600a554
Transfer-Encoding: chunked
X-Object-Meta-PIN: 2343
19
A bunch of data broken up
D
into chunks.
0
```

12.5.4.2. Copying Object

You can copy object from one container to another or add a new object and then add reference to designate the source of the data from another container.

To copy object from one container to another

- To add a new object and designate the source of the data from another container, run the following command:

```
COPY /<apiversion>/<account>/<container>/<sourceobject> HTTP/1.1
Host: <storage URL>
X-Auth-Token: < authentication-token-key>
Destination: /<container>/<destinationobject>
```

For example,

```
COPY /v1/AUTH_test/images/dogs HTTP/1.1
Host: example.storage.com
X-Auth-Token: AUTH_tkd3ad38b087b49bbbac0494f7600a554
Destination: /photos/cats

HTTP/1.1 201 Created
Date: Wed, 13 Jul 2011 18:32:21 GMT
Server: Apache
Content-Length: 0
Content-Type: text/plain; charset=UTF-8
```

To copy an object using cURL (for the above example), run the following command:

```
curl -v -X COPY -H 'X-Auth-Token:
AUTH_tkde3ad38b087b49bbbac0494f7600a554' -H 'Destination: /photos/cats' -k https://
example.storage.com:443/v1/AUTH_test/images/dogs
```

The status code of 201 (Created) indicates that you have successfully copied the object. If there is a missing content-Length or Content-Type header in the request, the status code of 412 (Length Required) is displayed.

You can also use PUT command to copy object by using additional header **`X-Copy-From: container/obj`**.

- To use PUT command to copy an object, run the following command:

```
PUT /v1/AUTH_test/photos/cats HTTP/1.1
Host: example.storage.com
X-Auth-Token: AUTH_tkd3ad38b087b49bbbac0494f7600a554
X-Copy-From: /images/dogs

HTTP/1.1 201 Created
Date: Wed, 13 Jul 2011 18:32:21 GMT
Server: Apache
Content-Type: text/plain; charset=UTF-8
```

To copy an object using cURL (for the above example), run the following command:

```
curl -v -X PUT -H 'X-Auth-Token: AUTH_tkde3ad38b087b49bbbac0494f7600a554'
-H 'X-Copy-From: /images/dogs' –k
https://example.storage.com:443/v1/AUTH_test/images/cats
```

The status code of 201 (Created) indicates that you have successfully copied the object.

12.5.4.3. Displaying Object Information

You can issue GET command on an object to view the object data of the object.

- To display the content of an object run the following command:

```
GET /<apiversion>/<account>/<container>/<object> HTTP/1.1
Host: <storage URL>
X-Auth-Token: <Authentication-token-key>
```

For example,

```
GET /v1/AUTH_test/images/cat HTTP/1.1
Host: example.storage.com
X-Auth-Token: AUTH_tkd3ad38b087b49bbbac0494f7600a554

HTTP/1.1 200 Ok
Date: Wed, 13 Jul 2011 23:52:21 GMT
Server: Apache
Last-Modified: Thu, 14 Jul 2011 13:40:18 GMT
ETag: 8a964ee2a5e88be344f36c22562a6486
Content-Length: 534210
[.........]
```

To display the content of an object using cURL (for the above example), run the following command:

```
curl -v -X GET -H 'X-Auth-Token:
AUTH_tkde3ad38b087b49bbbac0494f7600a554'
https://example.storage.com:443/v1/AUTH_test/images/cat -k
```

The status code of 200 (Ok) indicates the object#s data is displayed successfully. If that object does not exist, the status code 404 (Not Found) is displayed.

12.5.4.4. Displaying Object Metadata

You can issue HEAD command on an object to view the object metadata and other standard HTTP headers. You must send only authorization token as header.

- To display the metadata of the object, run the following command:

```
HEAD /<apiversion>/<account>/<container>/<object> HTTP/1.1
Host: <storage URL>
X-Auth-Token: <Authentication-token-key>
```

For example,

```
HEAD /v1/AUTH_test/images/cat HTTP/1.1
Host: example.storage.com
X-Auth-Token: AUTH_tkd3ad38b087b49bbbac0494f7600a554

HTTP/1.1 204 No Content
Date: Wed, 13 Jul 2011 21:52:21 GMT
Server: Apache
Last-Modified: Thu, 14 Jul 2011 13:40:18 GMT
ETag: 8a964ee2a5e88be344f36c22562a6486
Content-Length: 512000
Content-Type: text/plain; charset=UTF-8
X-Object-Meta-House: Cat
X-Object-Meta-Zoo: Cat
X-Object-Meta-Home: Cat
X-Object-Meta-Park: Cat
```

To display the metadata of the object using cURL (for the above example), run the following command:

```
curl -v -X HEAD -H 'X-Auth-Token:
AUTH_tkde3ad38b087b49bbbac0494f7600a554'
https://example.storage.com:443/v1/AUTH_test/images/cat -k
```

The status code of 204 (No Content) indicates the object#s metadata is displayed successfully. If that object does not exist, the status code 404 (Not Found) is displayed.

12.5.4.5. Updating Object Metadata

You can issue POST command on an object name only to set or overwrite arbitrary key metadata. You cannot change the object#s other headers such as Content-Type, ETag and others using POST operation. The POST command will delete all the existing metadata and replace it with the new arbitrary key metadata.

You must prefix **X-Object-Meta-** to the key names.

- To update the metadata of an object, run the following command:

  ```
  POST /<apiversion>/<account>/<container>/<object> HTTP/1.1
  Host: <storage URL>
  X-Auth-Token: <Authentication-token-key>
  X-Object-Meta-<key>: <new value>
  X-Object-Meta-<key>: <new value>
  ```

  For example,

  ```
  POST /v1/AUTH_test/images/cat HTTP/1.1
  Host: example.storage.com
  X-Auth-Token: AUTH_tkd3ad38b087b49bbbac0494f7600a554
  X-Object-Meta-Zoo: Lion
  X-Object-Meta-Home: Dog

  HTTP/1.1 202 Accepted
  Date: Wed, 13 Jul 2011 22:52:21 GMT
  Server: Apache
  Content-Length: 0
  Content-Type: text/plain; charset=UTF-8
  ```

  To update the metadata of an object using cURL (for the above example), run the following command:

  ```
  curl -v -X POST -H 'X-Auth-Token:
  AUTH_tkde3ad38b087b49bbbac0494f7600a554'
  https://example.storage.com:443/v1/AUTH_test/images/cat -H ' X-Object-
  Meta-Zoo: Lion' -H 'X-Object-Meta-Home: Dog' -k
  ```

  The status code of 202 (Accepted) indicates that you have successfully updated the object#s metadata. If that object does not exist, the status code 404 (Not Found) is displayed.

12.5.4.6. Deleting Object

You can use DELETE command to permanently delete the object.

The DELETE command on an object will be processed immediately and any subsequent operations like GET, HEAD, POST, or DELETE on the object will display 404 (Not Found) error.

- To delete an object, run the following command:

```
DELETE /<apiversion>/<account>/<container>/<object> HTTP/1.1
Host: <storage URL>
X-Auth-Token: <Authentication-token-key>
```

For example,

```
DELETE /v1/AUTH_test/pictures/cat HTTP/1.1
Host: example.storage.com
X-Auth-Token: AUTH_tkd3ad38b087b49bbbac0494f7600a554

HTTP/1.1 204 No Content
Date: Wed, 13 Jul 2011 20:52:21 GMT
Server: Apache
Content-Type: text/plain; charset=UTF-8
```

To delete an object using cURL (for the above example), run the following command:

```
curl -v -X DELETE -H 'X-Auth-Token:
AUTH_tkde3ad38b087b49bbbac0494f7600a554'
https://example.storage.com:443/v1/AUTH_test/pictures/cat -k
```

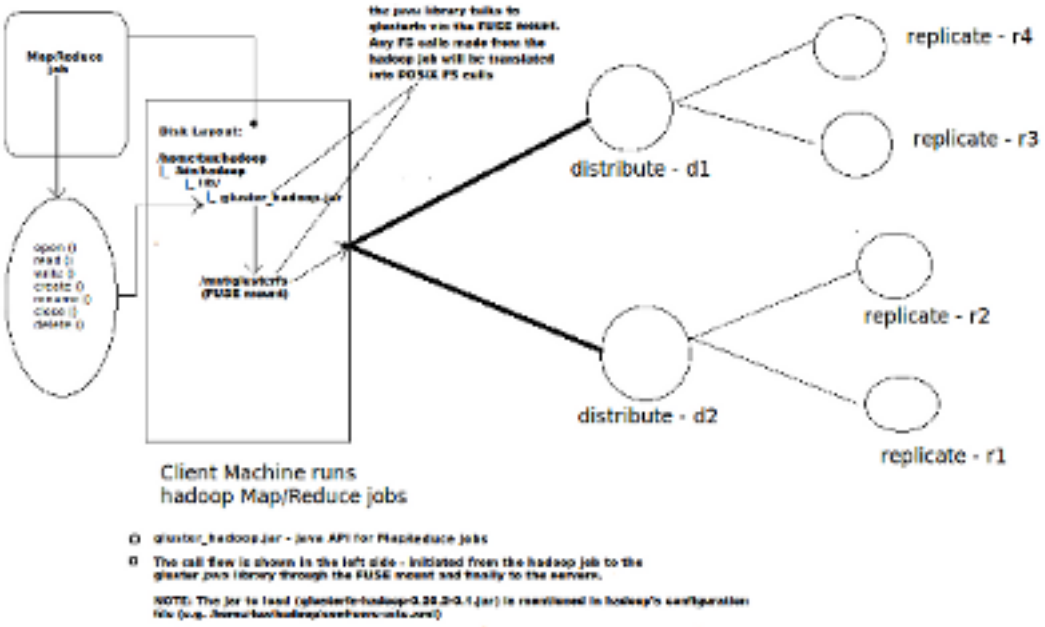
The status code of 204 (No Content) indicates that you have successfully deleted the object. If that object does not exist, the status code 404 (Not Found) is displayed.

Managing Hadoop Compatible Storage

GlusterFS provides compatibility for Apache Hadoop and it uses the standard file system APIs available in Hadoop to provide a new storage option for Hadoop deployments. Existing MapReduce based applications can use GlusterFS seamlessly. This new functionality opens up data within Hadoop deployments to any file-based or object-based application.

13.1. Architecture Overview

The following diagram illustrates Hadoop integration with GlusterFS:



13.2. Advantages

The following are the advantages of Hadoop Compatible Storage with GlusterFS:

- Provides simultaneous file-based and object-based access within Hadoop.
- Eliminates the centralized metadata server.
- Provides compatibility with MapReduce applications and rewrite is not required.
- Provides a fault tolerant file system.

13.3. Preparing to Install Hadoop Compatible Storage

This section provides information on pre-requisites and list of dependencies that will be installed during installation of Hadoop compatible storage.

13.3.1. Pre-requisites

The following are the pre-requisites to install Hadoop Compatible Storage :

- Hadoop 0.20.2 is installed, configured, and is running on all the machines in the cluster.
- Java Runtime Environment

- Maven (mandatory only if you are building the plugin from the source)
- JDK (mandatory only if you are building the plugin from the source)
- getfattr - command line utility

13.4. Installing, and Configuring Hadoop Compatible Storage

This section describes how to install and configure Hadoop Compatible Storage in your storage environment and verify that it is functioning correctly.

To install and configure Hadoop compatible storage:

1. Download **`glusterfs-hadoop-0.20.2-0.1.x86_64.rpm`** file to each server on your cluster. You can download the file from *http://download.gluster.com/pub/gluster/glusterfs/qa-releases/3.3-beta-2/glusterfs-hadoop-0.20.2-0.1.x86_64.rpm*.

2. To install Hadoop Compatible Storage on all servers in your cluster, run the following command:

   `# rpm –ivh --nodeps glusterfs-hadoop-0.20.2-0.1.x86_64.rpm`

   The following files will be extracted:

   - /usr/local/lib/glusterfs-*`Hadoop-version-gluster_plugin_version`*.jar
   - /usr/local/lib/conf/core-site.xml

3. (Optional) To install Hadoop Compatible Storage in a different location, run the following command:

   `# rpm –ivh --nodeps –prefix /usr/local/glusterfs/hadoop glusterfs-hadoop- 0.20.2-0.1.x86_64.rpm`

4. Edit the **`conf/core-site.xml`** file. The following is the sample **`conf/core-site.xml`** file:

```
<configuration>
  <property>
    <name>fs.glusterfs.impl</name>
    <value>org.apache.hadoop.fs.glusterfs.Gluster FileSystem</value>
</property>

<property>
   <name>fs.default.name</name>
   <value>glusterfs://fedora1:9000</value>
</property>

<property>
   <name>fs.glusterfs.volname</name>
   <value>hadoopvol</value>
</property>

<property>
   <name>fs.glusterfs.mount</name>
   <value>/mnt/glusterfs</value>
</property>

<property>
   <name>fs.glusterfs.server</name>
   <value>fedora2</value>
</property>
```

```
<property>
   <name>quick.slave.io</name>
   <value>Off</value>
</property>
</configuration>
```

The following are the configurable fields:

Property Name	Default Value	Description
fs.default.name	glusterfs://fedora1:9000	Any hostname in the cluster as the server and any port number.
fs.glusterfs.volname	hadoopvol	GlusterFS volume to mount.
fs.glusterfs.mount	/mnt/glusterfs	The directory used to fuse mount the volume.
fs.glusterfs.server	fedora2	Any hostname or IP address on the cluster except the client/master.
quick.slave.io	Off	Performance tunable option. If this option is set to On, the plugin will try to perform I/O directly from the disk file system (like ext3 or ext4) the file resides on. Hence read performance will improve and job would run faster. **Note** This option is not tested widely

5. Create a soft link in Hadoop's library and configuration directory for the downloaded files (in Step 3) using the following commands:

   # ln -s <*target location*> <*source location*>

   For example,

   # ln –s /usr/local/lib/glusterfs-0.20.2-0.1.jar *$HADOOP_HOME*/lib/glusterfs-0.20.2-0.1.jar

   # ln –s /usr/local/lib/conf/core-site.xml *$HADOOP_HOME*/conf/core-site.xml

6. (Optional) You can run the following command on Hadoop master to build the plugin and deploy it along with core-site.xml file, instead of repeating the above steps:

   # build-deploy-jar.py -d *$HADOOP_HOME* -c

13.5. Starting and Stopping the Hadoop MapReduce Daemon

To start and stop MapReduce daemon

- To start MapReduce daemon manually, enter the following command:

  ```
  # $HADOOP_HOME/bin/start-mapred.sh
  ```

- To stop MapReduce daemon manually, enter the following command:

  ```
  # $HADOOP_HOME/bin/stop-mapred.sh
  ```

Note

You must start Hadoop MapReduce daemon on all servers.

Troubleshooting GlusterFS

This section describes how to manage GlusterFS logs and most common troubleshooting scenarios related to GlusterFS.

14.1. Managing GlusterFS Logs

This section describes how to manage GlusterFS logs by performing the following operation:

- Rotating Logs

14.1.1. Rotating Logs

Administrators can rotate the log file in a volume, as needed.

To rotate a log file

- Rotate the log file using the following command:

  # gluster volume log rotate *VOLNAME*

  For example, to rotate the log file on test-volume:

```
# gluster volume log rotate test-volume
log rotate successful
```

Note

When a log file is rotated, the contents of the current log file are moved to log-file- name.epoch-time-stamp.

14.2. Troubleshooting Geo-replication

This section describes the most common troubleshooting scenarios related to GlusterFS Geo-replication.

14.2.1. Locating Log Files

For every Geo-replication session, the following three log files are associated to it (four, if the slave is a gluster volume):

- Master-log-file - log file for the process which monitors the Master volume
- Slave-log-file - log file for process which initiates the changes in slave
- Master-gluster-log-file - log file for the maintenance mount point that Geo-replication module uses to monitor the master volume
- Slave-gluster-log-file - is the slave's counterpart of it

Master Log File

To get the Master-log-file for geo-replication, use the following command:

gluster volume geo-replication MASTER SLAVE config log-file

For example:

 # gluster volume geo-replication Volume1 example.com:/data/remote_dir config log-file

Slave Log File

To get the log file for Geo-replication on slave (glusterd must be running on slave machine), use the following commands:

1. On master, run the following command:

   # gluster volume geo-replication Volume1 example.com:/data/remote_dir config session-owner 5f6e5200-756f-11e0-a1f0-0800200c9a66

   Displays the session owner details.

2. On slave, run the following command:

   # gluster volume geo-replication /data/remote_dir config log-file /var/log/gluster/${session-owner}:remote-mirror.log

3. Replace the session owner details (output of Step 1) to the output of the Step 2 to get the location of the log file.

   /var/log/gluster/5f6e5200-756f-11e0-a1f0-0800200c9a66:remote-mirror.log

14.2.2. Rotating Geo-replication Logs

Administrators can rotate the log file of a particular master-slave session, as needed. When you run geo-replication's **log-rotate** command, the log file is backed up with the current timestamp suffixed to the file name and signal is sent to gsyncd to start logging to a new log file.

To rotate a geo-replication log file

- Rotate log file for a particular master-slave session using the following command:

  # gluster volume geo-replication *master slave* log-rotate

  For example, to rotate the log file of master **Volume1** and slave **example.com:/data/remote_dir** :

  ```
  # gluster volume geo-replication Volume1 example.com:/data/remote_dir log rotate
  log rotate successful
  ```

- Rotate log file for all sessions for a master volume using the following command:

  # gluster volume geo-replication *master* log-rotate

  For example, to rotate the log file of master **Volume1**:

  ```
  # gluster volume geo-replication Volume1 log rotate
  log rotate successful
  ```

- Rotate log file for all sessions using the following command:

  `# gluster volume geo-replication log-rotate`

  For example, to rotate the log file for all sessions:

```
# gluster volume geo-replication log-rotate
log rotate successful
```

14.2.3. Synchronization is not complete

Description: GlusterFS Geo-replication did not synchronize the data completely but still the geo-replication status displayed is OK.

Solution: You can enforce a full sync of the data by erasing the index and restarting GlusterFS Geo-replication. After restarting, GlusterFS Geo-replication begins synchronizing all the data. All files are compared using checksum, which can be a lengthy and high resource utilization operation on large data sets. If the error situation persists, contact Red Hat Support.

For more information about erasing index, see *Section 7.1, “Tuning Volume Options”*.

14.2.4. Issues in Data Synchronization

Description: Geo-replication display status as OK, but the files do not get synced, only directories and symlink gets synced with the following error message in the log:

[2011-05-02 13:42:13.467644] E [master:288:regjob] GMaster: failed to sync ./some_file`

Solution: Geo-replication invokes rsync v3.0.0 or higher on the host and the remote machine. You must verify if you have installed the required version.

14.2.5. Geo-replication status displays Faulty very often

Description: Geo-replication displays status as faulty very often with a backtrace similar to the following:

2011-04-28 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 error indicates that the RPC communication between the master geo-replication module and slave geo-replication module is broken and this can happen for various reasons. Check if it satisfies all the following pre-requisites:

- Password-less SSH is set up properly between the host where geo-replication command is executed and the remote machine where the slave geo-replication is located.
- If FUSE is installed in the machine where the geo-replication command is executed, because geo-replication module mounts the GlusterFS volume using FUSE to sync data.
- If the **Slave** is a volume, check if that volume is started.
- If the Slave is a plain directory, verify if the directory has been created already with the required permissions.

- If GlusterFS 3.3 or higher is not installed in the default location (in Master) and has been prefixed to be installed in a custom location, configure the **`gluster-command`** for it to point to the exact location.
- If GlusterFS 3.3 or higher is not installed in the default location (in slave) and has been prefixed to be installed in a custom location, configure the **`remote-gsyncd-command`** for it to point to the exact place where geo-replication is located.

14.2.6. Intermediate Master goes to Faulty State

Description: In a cascading set-up, the intermediate master goes to faulty state with the following 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 set-up the Intermediate master is loyal to the original primary master. The above log means that the geo-replication module has detected change in primary master. If this is the desired behavior, delete the config option volume-id in the session initiated from the intermediate master.

14.3. Troubleshooting POSIX ACLs

This section describes the most common troubleshooting issues related to POSIX ACLs.

14.3.1. setfacl command fails with “setfacl: <file or directory name>: Operation not supported” error

You may face this error when the backend file systems in one of the servers is not mounted with the "-o acl" option. The same can be confirmed by viewing the following error message in the log file of the server "Posix access control list is not supported".

Solution: Remount the backend file system with "-o acl" option. For more information, see *Section 11.1.1, “Activating POSIX ACLs Support on Sever ”*.

14.4. Troubleshooting Hadoop Compatible Storage

This section describes the most common troubleshooting issues related to Hadoop Compatible Storage.

14.4.1. Time Sync

Running MapReduce job may throw exceptions if the clocks are out-of-sync on the hosts in the cluster.

Solution: Sync the time on all hosts using ntpd program.

14.5. Troubleshooting NFS

This section describes the most common troubleshooting issues related to NFS .

14.5.1. mount command on NFS client fails with “RPC Error: Program not registered”

Start portmap or rpcbind service on the machine where NFS server is running.

This error is encountered when the server has not started correctly.

On most Linux distributions this is fixed by starting portmap:

```
$ /etc/init.d/portmap start
```

On some distributions where portmap has been replaced by rpcbind, the following command is required:

```
$ /etc/init.d/rpcbind start
```

After starting portmap or rpcbind, gluster NFS server needs to be restarted.

14.5.2. NFS server start-up fails with “Port is already in use” error in the log file."

Another Gluster NFS server is running on the same machine.

This error can arise in case there is already a Gluster 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
```

To resolve this error one of the Gluster NFS servers will have to be shutdown. At this time, Gluster NFS server does not support running multiple NFS servers on the same machine.

14.5.3. mount command fails with “rpc.statd” related error message

If the mount command fails with the following error message:

mount.nfs: rpc.statd is not running but is required for remote locking. mount.nfs: Either use '-o nolock' to keep locks local, or start statd.

Start rpc.statd

For NFS clients to mount the NFS server, rpc.statd service must be running on the client machine.

Start rpc.statd service by running the following command:

```
$ rpc.statd
```

14.5.4. mount command takes too long to finish.

Start rpcbind service on the NFS client.

The problem is that the rpcbind or portmap service is not running on the NFS client. The resolution for this is to start either of these services by running the following command:

```
$ /etc/init.d/portmap start
```

On some distributions where portmap has been replaced by rpcbind, the following command is required:

```
$ /etc/init.d/rpcbind start
```

14.5.5. NFS server, glusterfsd starts but initialization fails with “nfsrpc- service: portmap registration of program failed” error message in the log.

NFS start-up can succeed 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 portmap or rpcbind service on the NFS server.

   On most Linux distributions, portmap can be started using the following command:

   ```
   $ /etc/init.d/portmap start
   ```

   On some distributions where portmap has been replaced by rpcbind, run the following command:

   ```
   $ /etc/init.d/rpcbind start
   ```

   After starting portmap or rpcbind, gluster 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 Gluster NFS server. On Linux systems, this could be the kernel NFS server. Resolution involves stopping the other NFS server or not running the Gluster 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:

   ```
   $ /etc/init.d/nfs-kernel-server stop
   ```

   ```
   $ /etc/init.d/nfs stop
   ```

3. Restart Gluster NFS server.

14.5.6. mount command fails with NFS server failed error.

mount command fails with following error

mount: mount to NFS server '10.1.10.11' failed: timed out (retrying).

Perform one of the following to resolve this issue:

1. 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 hostnames 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 responsd to DNS request. These delays can result in delayed replies from the NFS server to the NFS client resulting in the timeout error seen above.

   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 rpc-auth.addr.namelookup off
   ```

   Note

   Note: Remember that disabling the NFS server forces authentication of clients to use only IP addresses and if the authentication rules in the volume file use hostnames, those authentication rules will fail and disallow mounting for those clients.

   or

2. NFS version used by the NFS client is other than version 3.

   Gluster NFS server supports version 3 of NFS protocol. 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 Gluster NFS server because it is using version 4 messages which are not understood by Gluster 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 mount-point
   ```

14.5.7. showmount fails with clnt_create: RPC: Unable to receive

Check your firewall setting to open ports 111 for portmap requests/replies and Gluster NFS server requests/replies. Gluster NFS server operates over the following port numbers: 38465, 38466, and 38467.

For more information, see *Section 6.1.1.1, “Installing on RPM Based Distributions ”*.

14.5.8. Application fails with "Invalid argument" or "Value too large for defined data type" error.

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 CLI to make Gluster NFS server return 32-bit inode numbers instead: nfs.enable-ino32 <on|off>

Applications that will benefit are those that were either:

- built 32-bit and run on 32-bit machines such that they do not support large files by default
- built 32-bit on 64-bit systems

This option is disabled by default. So Gluster NFS server returns 64-bit inode numbers by default.

Applications which can be rebuilt from source are recommended to rebuild using the following flag with gcc:

-D_FILE_OFFSET_BITS=64

14.6. Troubleshooting File Locks

In GlusterFS 3.3 you can use **statedump** command to list the locks held on files. The statedump output also provides information on each lock with its range, basename, PID of the application holding the lock, and so on. You can analyze the output to know about the locks whose owner/application is no longer running or interested in that lock. After ensuring that the no application is using the file, you can clear the lock using the following **clear lock** 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 10.4, "Performing Statedump on a Volume "*

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* inode

   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 **server.statedump-path** volume option. The naming convention of the dump file is **<brick-path>.<brick-pid>.dump**.

   The following are the sample contents of the statedump file. It indicates that GlusterFS has entered into a state where there is an entry lock (entrylk) and an inode lock (inodelk). Ensure that those are stale locks and no resources own them before clearing.

   ```
   [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./gfs/brick1.hashsize=14057
   conn.2.bound_xl./gfs/brick1.name=/gfs/brick1/inode
   conn.2.bound_xl./gfs/brick1.lru_limit=16384
   conn.2.bound_xl./gfs/brick1.active_size=2
   ```

```
conn.2.bound_xl./gfs/brick1.lru_size=0
conn.2.bound_xl./gfs/brick1.purge_size=0

[conn.2.bound_xl./gfs/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
```

2. Clear the lock using the following command:

   # gluster volume clear-locks *VOLNAME path* kind granted entry basename

   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
   vol-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

   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
   vol-locks: inode blocked locks=0 granted locks=1
   ```

   You can perform statedump on test-volume again to verify that the above inode and entry locks are cleared.

Command Reference

This section describes the available commands and includes thefollowing section:

- gluster Command

  Gluster Console Manager (command line interpreter)

- glusterd Daemon

  Gluster elastic volume management daemon

15.1. gluster Command

NAME

gluster - Gluster Console Manager (command line interpreter)

SYNOPSIS

To run the program and display the gluster prompt:

gluster

To specify a command directly: gluster [COMMANDS] [OPTIONS]

DESCRIPTION

The Gluster Console Manager is a command line utility for elastic volume management. 'gluster' command enables administrators to perform cloud operations such as creating, expanding, shrinking, rebalancing, and migrating volumes without needing to schedule server downtime.

COMMANDS

Command	Description
Volume	
volume info [all | VOLNAME]	Displays information about all volumes, or the specified volume.
volume create NEW-VOLNAME [stripe COUNT] [replica COUNT] [transport tcp | rdma | tcp,rdma] NEW-BRICK ...	Creates a new volume of the specified type using the specified bricks and transport type (the default transport type is tcp). NOTE: with 3.3.0 release, transport type 'rdma' and 'tcp,rdma' are not fully supported.
volume delete VOLNAME	Deletes the specified volume.
volume start VOLNAME	Starts the specified volume.
volume stop VOLNAME [force]	Stops the specified volume.
volume help	Displays help for the volume command.
Brick	
volume add-brick VOLNAME [replica N] [stripe N] NEW-BRICK1 ...	Adds the specified brick(s) to the given VOLUME. Using add-brick, users can increase the replica/stripe count of the volume, or increase the volume capacity by adding the brick(s) without changing volume type.
volume replace-brick VOLNAME (BRICK NEW-BRICK) [start | start force |	Used to replace BRICK with NEW-BRICK in a given VOLUME. After replace-brick is complete, the changes to get reflected in volume information, replace-brick 'commit' command is neccessary.

Command	Description
abort | status | commit | commit force]	
volume remove-brick VOLNAME [replica N] BRICK1 ... [start | stop | status | commit | force]	Removes the specified brick(s) from the specified volume. 'remove-brick' command can be used to reduce the replica count of the volume when 'replica N' option is given. To ensure data migration from the removed brick to existing bricks, give 'start' sub-command at the end of the command. After the 'status' command says remove-brick operation is complete, user can 'commit' the changes to volume file. Using 'remove-brick' without 'start' option works similar to 'force' command, which makes the changes to volume configuration without migrating the data.
Rebalance	
volume rebalance VOLNAME start	Starts rebalancing of the data on specified volume.
volume rebalance VOLNAME stop	Stops rebalancing the specified volume.
volume rebalance VOLNAME status	Displays the rebalance status of the specified volume.
Log	
volume log rotate VOLNAME [BRICK]	Rotates the log file for corresponding volume/brick.
Debugging	
volume top VOLNAME {[open|read|write|opendir|readdir [nfs]] |[read-perf|write-perf [nfs|{bs COUNT count COUNT}]]|[clear [nfs]]} [BRICK] [list-cnt COUNT]	Shows the operation details on the volume depending on the arguments given.
volume profile VOLNAME {start|info|stop} [nfs]	Shows the file operation details on each bricks of the volume.
volume status [all | VOLNAME] [nfs|shd|BRICK] [detail|clients|mem|inode|fd|callpool]	Show details of activity, internal data of the processes (nfs/shd/BRICK) corresponding to one of the next argument given. If now argument is given, this command outputs bare minimum details of the current status (include PID of brick process etc) of volume's bricks.
statedump VOLNAME [nfs] [all|mem|iobuf|callpool|priv|fd|inode|history]	Command is used to take the statedump of the process, which is used captures most of the internal details.
Peer	
peer probe HOSTNAME	Probes the specified peer.
peer detach HOSTNAME	Detaches the specified peer.
peer status	Displays the status of peers.
peer help	Displays help for the peer command.
Geo-replication	

Command	Description	
volume geo-replication MASTER SLAVE start	Start geo-replication between the hosts specified by MASTER and SLAVE. You can specify a local master volume as :VOLNAME. You can specify a local slave volume as :VOLUME and a local slave directory as /DIRECTORY/SUB-DIRECTORY. You can specify a remote slave volume as DOMAIN::VOLNAME and a remote slave directory as DOMAIN:/DIRECTORY/SUB-DIRECTORY.	
volume geo-replication MASTER SLAVE stop	Stop geo-replication between the hosts specified by MASTER and SLAVE. You can specify a local master volume as :VOLNAME and a local master directory as /DIRECTORY/SUB-DIRECTORY. You can specify a local slave volume as :VOLNAME and a local slave directory as /DIRECTORY/SUB-DIRECTORY. You can specify a remote slave volume as DOMAIN::VOLNAME and a remote slave directory as DOMAIN:/DIRECTORY/SUB-DIRECTORY.	
volume geo-replication MASTER SLAVE config [options]	Configure geo-replication options between the hosts specified by MASTER and SLAVE.	
	gluster-command COMMAND	The path where the gluster command is installed.
	gluster-log-level LOGFILELEVEL	The log level for gluster processes.
	log-file LOGFILE	The path to the geo-replication log file.
	log-level LOGFILELEVEL	The log level for geo-replication.
	remote-gsyncd COMMAND	The path where the gsyncd binary is installed on the remote machine.
	ssh-command COMMAND	The ssh command to use 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).
	volume_id= UID	The command to delete the existing master UID for the intermediate/slave node.
	timeout SECONDS	The timeout period.
	sync-jobs N	The number of simultaneous files/directories that can be synchronized.

Command	Description	
ignore-deletes	If this option is set to 1, a file deleted on master will not trigger a delete operation on the slave. Hence, the slave will remain as a superset of the master and can be used to recover the master in case of crash and/or accidental delete.	
Other		
help	Display the command options.	
quit		Exit the gluster command line interface.

FILES

/var/lib/glusterd/*

15.2. glusterd Daemon

NAME

glusterd - Gluster elastic volume management daemon

SYNOPSIS

glusterd [OPTION...]

DESCRIPTION

The glusterd daemon is used for elastic volume management. The daemon must be run on all export servers.

OPTIONS

Option	Description
Basic	
-l=LOGFILE, --log-file=LOGFILE	Files to use for logging (the default is /usr/local/var/log/glusterfs/glusterfs.log).
-L=LOGLEVEL, --log-level=LOGLEVEL	Logging severity. Valid options are TRACE, DEBUG, INFO, WARNING, ERROR and CRITICAL (the default is INFO).
--debug	Runs the program in debug mode. This option sets --no-daemon, --log-level to DEBUG, and --log-file to console.
-N, --no-daemon	Runs the program in the foreground.
Miscellaneous	
-?, --help	Displays this help.
--usage	Displays a short usage message.
-V, --version	Prints the program version.

FILES

/var/lib/glusterd/*

Glossary

Brick	A Brick is 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/**
Cluster	A cluster is a group of linked computers, working together closely thus in many respects forming a single computer.
Distributed File System	A file system that allows multiple clients to concurrently access data over a computer network.
Filesystem	A method of storing and organizing computer files and their data. Essentially, it organizes these files into a database for the storage, organization, manipulation, and retrieval by the computer's operating system. Source: *Wikipedia*[1]
FUSE	Filesystem in Userspace (FUSE) is a loadable kernel module for Unix-like computer 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 actual kernel interfaces. Source: *Wikipedia*[2]
Geo-Replication	Geo-replication provides a continuous, asynchronous, and incremental replication service from site to another over Local Area Networks (LAN), Wide Area Network (WAN), and across the Internet.
glusterd	The Gluster management daemon that needs to run on all servers in the trusted storage pool.
Metadata	Metadata is data providing information about one or more other pieces of data.
Namespace	Namespace is an abstract container or environment created to hold a logical grouping of unique identifiers or symbols. Each Gluster volume exposes a single namespace as a POSIX mount point that contains every file in the cluster.
Open Source	Open source describes practices in production and development that promote access to the end product's source materials. Some consider open source a philosophy, others consider it a pragmatic methodology.

[1] http://en.wikipedia.org/wiki/Filesystem
[2] http://en.wikipedia.org/wiki/Filesystem_in_Userspace

	Before the term open source became widely adopted, developers and producers used a variety of phrases to describe the concept; open source gained hold with the rise of the Internet, and the attendant need for massive retooling of the computing source code. Opening the source code enabled a self-enhancing diversity of production models, communication paths, and interactive communities. Subsequently, a new, three-word phrase "open source software" was born to describe the environment that the new copyright, licensing, domain, and consumer issues created. Source: *Wikipedia*[3]
Petabyte	A petabyte (derived from the SI prefix peta-) is a unit of information equal to one quadrillion (short scale) 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 = 10005 B = 1015 B. The term "pebibyte" (PiB), using a binary prefix, is used for the corresponding power of 1024. Source: *Wikipedia*[4]
POSIX	Portable Operating System Interface (for Unix) is the name of a family of related standards specified by the IEEE to define the application programming interface (API), along with shell and utilities interfaces for software compatible with variants of the Unix operating system. Gluster exports a fully POSIX compliant file system.
RAID	Redundant Array of Inexpensive Disks (RAID) is a technology that provides increased storage reliability through redundancy, combining 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 A records with the same name and different IP addresses in the zone file of a DNS server.
Trusted Storage Pool	A storage pool is a trusted network of storage servers. When you start the first server, the storage pool consists of that server alone.
Userspace	Applications running in user space don't directly interact with hardware, instead using the kernel to moderate access. Userspace applications are generally more portable than applications in kernel space. Gluster is a user space application.
Volfile	Volfile is a configuration file used by glusterfs process. Volfile will be usually located at **`/etc/glusterd/vols/VOLNAME`**.

[3] http://en.wikipedia.org/wiki/Open_source
[4] http://en.wikipedia.org/wiki/Petabyte

Volume	A volume is a logical collection of bricks. Most of the gluster management operations happen on the volume.

Appendix A. Revision History

Revision 1-0 **Thu Apr 5 2012** **Divya Muntimadugu** *divya@redhat.com*

Draft