Red Hat Storage 2.0
Administration Guide

Configuring and Managing Red Hat Storage Server
Edition 1

Divya Muntimadugu    Anjana Suparna Sriram
Red Hat Storage 2.0 Administration Guide

Configuring and Managing Red Hat Storage Server Edition 1

Divya Muntimadugu
Red Hat Engineering Content Services
divya@redhat.com

Anjana Suparna Sriram
Red Hat Engineering Content Services
asriram@redhat.com
Legal Notice

Copyright © 2013 Red Hat Inc.

This document is licensed by Red Hat under the Creative Commons Attribution-ShareAlike 3.0 Unported License. If you distribute this document, or a modified version of it, you must provide attribution to Red Hat, Inc. and provide a link to the original. If the document is modified, all Red Hat trademarks must be removed.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, JBoss, MetaMatrix, Fedora, the Infinity Logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux® is the registered trademark of Linus Torvalds in the United States and other countries.

Java® is a registered trademark of Oracle and/or its affiliates.

XFS® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js® is an official trademark of Joyent. Red Hat Software Collections is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack® Word Mark and OpenStack Logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

Abstract

Red Hat Storage Administration Guide describes the configuration and management of Red Hat Storage Server for On-Premise and Public Cloud.
# Table of Contents

**Preface** .............................................................................................................................. 7  
1. Audience 7  
2. License 7  
3. Document Conventions 7  
   3.1. Typographic Conventions 7  
   3.2. Pull-quote Conventions 9  
   3.3. Notes and Warnings 9  
4. Getting Help and Giving Feedback 10  
   4.1. Do You Need Help? 10  
   4.2. We Need Feedback! 10  

**Part I. Introduction** ............................................................................................................. 11  

**Chapter 1. Introducing Red Hat Storage** ............................................................................. 12  

**Chapter 2. Red Hat Storage Architecture** ......................................................................... 13  
2.1. Red Hat Storage Server for On-Premise Architecture 13  
2.2. Red Hat Storage Server for Public Cloud 14  

**Chapter 3. Key Features** ..................................................................................................... 16  
3.1. Elasticity 16  
3.2. No Metadata with the Elastic Hash Algorithm 16  
3.3. Scalability 16  
3.4. High Availability and Flexibility 16  
3.5. Flexibility 16  
3.6. No Application Rewrites 16  
3.7. Simple Management 17  
3.8. Modular, Stackable Design 17  
3.9. Unified File and Object Storage 17  
3.10. Hadoop Compatible Storage 17  

**Chapter 4. Use Case Examples** .......................................................................................... 18  
4.1. Use Case 1: Using Red Hat Storage for Data Archival 18  
   4.1.1. Key Features of Red Hat Storage Server for Nearline and Archival Use Case 19  
4.2. Use Case 2: Using Red Hat Storage for High Performance Computing 19  
   4.2.1. Key Features of Red Hat Storage Server for High Performance Computing Use Case 20  
4.3. Use Case 3: Using Red Hat Storage for Content Clouds 21  
   4.3.1. Key Features of Red Hat Storage Server for Content Clouds Use Case 21  

**Chapter 5. Storage Concepts** ............................................................................................. 23  

**Part II. Red Hat Storage Administration On-Premise** ......................................................... 26  

**Chapter 6. Managing the glusterd Service** ......................................................................... 27  
6.1. Starting and Stopping glusterd Manually 27  

**Chapter 7. Setting up Trusted Storage Pools** .................................................................... 28  
7.1. Adding Servers to Trusted Storage Pool 28  
7.2. Removing Servers from the Trusted Storage Pool 29  

**Chapter 8. Setting up Red Hat Storage Volumes** ............................................................... 30  
8.1. Formatting and Mounting Bricks 31  
8.2. Encrypted Disk 32  
8.3. Creating Distributed Volumes 32
# Red Hat Storage 2.0 Administration Guide

## 8. Creating Volumes

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.4. Creating Replicated Volumes</td>
<td>34</td>
</tr>
<tr>
<td>8.5. Creating Distributed Replicated Volumes</td>
<td>35</td>
</tr>
<tr>
<td>8.6. Creating Striped Volumes</td>
<td>37</td>
</tr>
<tr>
<td>8.7. Creating Striped Replicated Volumes</td>
<td>39</td>
</tr>
<tr>
<td>8.8. Creating Distributed Striped Volumes</td>
<td>40</td>
</tr>
<tr>
<td>8.9. Creating Distributed Striped Replicated Volumes</td>
<td>41</td>
</tr>
<tr>
<td>8.10. Starting Volumes</td>
<td>43</td>
</tr>
</tbody>
</table>

## 9. Accessing Data - Setting Up Clients

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.1. Native Client</td>
<td>44</td>
</tr>
<tr>
<td>9.1.1. Installing Native Client</td>
<td>44</td>
</tr>
<tr>
<td>9.1.2. Mounting Red Hat Storage Volumes</td>
<td>45</td>
</tr>
<tr>
<td>9.1.2.1. Manually Mounting Volumes</td>
<td>46</td>
</tr>
<tr>
<td>9.1.2.2. Automatically Mounting Volumes</td>
<td>47</td>
</tr>
<tr>
<td>9.1.2.3. Testing Mounted Volumes</td>
<td>47</td>
</tr>
<tr>
<td>9.2. NFS</td>
<td>48</td>
</tr>
<tr>
<td>9.2.1. Using NFS to Mount Red Hat Storage Volumes</td>
<td>48</td>
</tr>
<tr>
<td>9.2.1.1. Manually Mounting Volumes Using NFS</td>
<td>48</td>
</tr>
<tr>
<td>9.2.1.2. Automatically Mounting Volumes Using NFS</td>
<td>49</td>
</tr>
<tr>
<td>9.2.1.3. Testing Volumes Mounted Using NFS</td>
<td>50</td>
</tr>
<tr>
<td>9.2.2. Troubleshooting NFS</td>
<td>50</td>
</tr>
<tr>
<td>9.2.2.1. mount command on NFS client fails with &quot;RPC Error: Program not registered&quot;</td>
<td>50</td>
</tr>
<tr>
<td>9.2.2.2. NFS server glusterfsd starts but initialization fails with “nfsrpc- service: portmap registration of program failed” error message in the log.</td>
<td>51</td>
</tr>
<tr>
<td>9.2.2.3. NFS server start-up fails with &quot;Port is already in use&quot; error in the log file.&quot;</td>
<td>51</td>
</tr>
<tr>
<td>9.2.2.4. mount command takes too long to finish.</td>
<td>52</td>
</tr>
<tr>
<td>9.2.2.5. mount command fails with NFS server failed error.</td>
<td>52</td>
</tr>
<tr>
<td>9.2.2.6. showmount fails with clnt_create: RPC: Unable to receive</td>
<td>53</td>
</tr>
<tr>
<td>9.2.2.7. Application fails with &quot;Invalid argument&quot; or &quot;Value too large for defined data type&quot; error.</td>
<td>53</td>
</tr>
<tr>
<td>9.3. SMB</td>
<td>53</td>
</tr>
<tr>
<td>9.3.1. Mounting Red Hat Storage Volumes as SMB Shares</td>
<td>54</td>
</tr>
<tr>
<td>9.3.1.1. Exporting Red Hat Storage Volumes Through Samba</td>
<td>54</td>
</tr>
<tr>
<td>9.3.1.2. Automatically Exporting Red Hat Storage Volumes Through Samba</td>
<td>54</td>
</tr>
<tr>
<td>9.3.1.3. Manually Mounting Volumes Using SMB on Red Hat Enterprise Linux and Windows</td>
<td>55</td>
</tr>
<tr>
<td>9.3.1.4. Automatically Mounting Volumes Using SMB on Red Hat Enterprise Linux and Windows</td>
<td>55</td>
</tr>
<tr>
<td>9.3.1.5. Testing Volumes Mounted Using SMB on Red Hat Enterprise Linux and Windows</td>
<td>55</td>
</tr>
<tr>
<td>9.4. Configuring Automated IP Failover for NFS and SMB</td>
<td>56</td>
</tr>
<tr>
<td>9.4.1. Setting Up CTDB</td>
<td>57</td>
</tr>
<tr>
<td>9.4.2. Starting and Verifying your Configuration</td>
<td>58</td>
</tr>
<tr>
<td>9.5. POSIX Access Control Lists</td>
<td>59</td>
</tr>
<tr>
<td>9.5.1. Setting POSIX ACLs</td>
<td>59</td>
</tr>
<tr>
<td>9.5.1.1. Setting Access ACLs</td>
<td>59</td>
</tr>
<tr>
<td>9.5.1.2. Setting Default ACLs</td>
<td>60</td>
</tr>
<tr>
<td>9.5.2. Retrieving POSIX ACLs</td>
<td>60</td>
</tr>
<tr>
<td>9.5.3. Removing POSIX ACLs</td>
<td>61</td>
</tr>
<tr>
<td>9.5.4. Samba and ACLs</td>
<td>61</td>
</tr>
<tr>
<td>9.5.5. NFS and ACLs</td>
<td>61</td>
</tr>
</tbody>
</table>

## 10. Managing Red Hat Storage Volumes

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.1. Tuning Volume Options</td>
<td>62</td>
</tr>
<tr>
<td>10.2. Expanding Volumes</td>
<td>66</td>
</tr>
<tr>
<td>10.3. Shrinking Volumes</td>
<td>67</td>
</tr>
<tr>
<td>10.3.1. Stopping Remove Brick Operation</td>
<td>69</td>
</tr>
<tr>
<td>10.4. Migrating Volumes</td>
<td>69</td>
</tr>
<tr>
<td>Chapter 19. Managing Hadoop Compatible Storage</td>
<td>158</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-----</td>
</tr>
<tr>
<td>19.1. Architecture Overview</td>
<td>158</td>
</tr>
<tr>
<td>19.2. Advantages</td>
<td>158</td>
</tr>
<tr>
<td>19.3. Preparing to Install Hadoop Compatible Storage</td>
<td>159</td>
</tr>
<tr>
<td>19.3.1. Pre-requisites</td>
<td>159</td>
</tr>
<tr>
<td>19.4. Configuring Hadoop Compatible Storage</td>
<td>159</td>
</tr>
<tr>
<td>19.5. Starting and Stopping the Hadoop MapReduce Daemon</td>
<td>161</td>
</tr>
<tr>
<td>19.6. Troubleshooting Hadoop Compatible Storage</td>
<td>162</td>
</tr>
<tr>
<td>19.6.1. Time Sync</td>
<td>162</td>
</tr>
</tbody>
</table>

| Part V. Appendices                          | 163 |

<table>
<thead>
<tr>
<th>Chapter 20. Command Reference</th>
<th>164</th>
</tr>
</thead>
<tbody>
<tr>
<td>20.1. gluster Command</td>
<td>164</td>
</tr>
<tr>
<td>20.2. glusterd Daemon</td>
<td>169</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 21. Troubleshooting</th>
<th>170</th>
</tr>
</thead>
<tbody>
<tr>
<td>21.1. Managing Red Hat Storage Logs</td>
<td>170</td>
</tr>
<tr>
<td>21.1.1. Rotating Logs</td>
<td>170</td>
</tr>
<tr>
<td>21.2. Troubleshooting File Locks</td>
<td>170</td>
</tr>
</tbody>
</table>

| Revision History                           | 175 |
Preface

Red Hat Storage is scale-out network attached storage (NAS) for private cloud or datacenter, public cloud, and hybrid cloud environments. It is software-only, open source, and designed to meet unstructured data storage requirements.

This guide introduces Red Hat Storage, describes the minimum requirements, and provides step-by-step instructions to install the software and manage your storage environment.

1. Audience

The Red Hat Storage Administrator’s Guide is intended for system and storage administrators who need to configure, administer, manage Red Hat Storage Server. To use this guide, you should be familiar with basic Linux operating system concepts and administrative procedures, concepts of file system, and Storage concepts.

If you want to install Red Hat Storage Server, read the Red Hat Storage Installation Guide.

2. License

The Red Hat Storage End User License Agreement (EULA) 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 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 include 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 keys 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 key, all presented in mono-spaced bold and all distinguishable thanks to context.

Key combinations can be distinguished from an individual key by the plus sign that connects each part of a key combination. For example:

Press Enter to execute the command.
Press \texttt{Ctrl+Alt+F2} to switch to a virtual terminal.

The first example highlights a particular key to press. The second example highlights a key combination: a set of three keys pressed simultaneously.

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

\begin{itemize}
  \item File-related classes include \texttt{filesystem} for file systems, \texttt{file} for files, and \texttt{dir} for directories. Each class has its own associated set of permissions.
\end{itemize}

\begin{itemize}
  \item Proportional Bold
\end{itemize}

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:

\begin{itemize}
  \item Choose \texttt{System $\rightarrow$ Preferences $\rightarrow$ Mouse} from the main menu bar to launch \texttt{Mouse Preferences}. In the \texttt{Buttons} tab, select the \texttt{Left-handed mouse} check box and click \texttt{Close} to switch the primary mouse button from the left to the right (making the mouse suitable for use in the left hand).

  \item To insert a special character into a \texttt{gedit} file, choose \texttt{Applications $\rightarrow$ Accessories $\rightarrow$ Character Map} from the main menu bar. Next, choose \texttt{Search $\rightarrow$ Find…} from the \texttt{Character Map} menu bar, type the name of the character in the \texttt{Search} field and click \texttt{Next}. The character you sought will be highlighted in the \texttt{Character Table}. Double-click this highlighted character to place it in the \texttt{Text to copy} field and then click the \texttt{Copy} button. Now switch back to your document and choose \texttt{Edit $\rightarrow$ Paste} from the \texttt{gedit} menu bar.
\end{itemize}

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.

\begin{itemize}
  \item Mono-spaced Bold Italic or Proportional Bold Italic
\end{itemize}

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:

\begin{itemize}
  \item To connect to a remote machine using ssh, type \texttt{ssh username@domain.name} at a shell prompt. If the remote machine is \texttt{example.com} and your username on that machine is john, type \texttt{ssh john@example.com}.

  \item The \texttt{mount -o remount file-system} command remounts the named file system. For example, to remount the \texttt{/home} file system, the command is \texttt{mount -o remount /home}.

  \item To see the version of a currently installed package, use the \texttt{rpm -q package} command. It will return a result as follows: \texttt{package-version-release}.
\end{itemize}

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:

<table>
<thead>
<tr>
<th>books</th>
<th>Desktop</th>
<th>documentation</th>
<th>drafts</th>
<th>mss</th>
<th>photos</th>
<th>stuff</th>
<th>svn</th>
</tr>
</thead>
<tbody>
<tr>
<td>books_tests</td>
<td>Desktop1</td>
<td>downloads</td>
<td>images</td>
<td>notes</td>
<td>scripts</td>
<td>svgs</td>
<td></td>
</tr>
</tbody>
</table>

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

```c
static int kvm_vm_ioctl_deassign_device(struct kvm *kvm,
                                        struct kvm_assigned_pci_dev *assigned_dev)
{
    int r = 0;
    struct kvm_assigned_dev_kernel *match;

    mutex_lock(&kvm->lock);

    match = kvm_find_assigned_dev(&kvm->arch.assigned_dev_head,
                                  assigned_dev->assigned_dev_id);
    if (!match) {
        printk(KERN_INFO "%s: device hasn't been assigned before, "
                "so cannot be deassigned\n", __func__);
        r = -EINVAL;
        goto out;
    }

    kvm_deassign_device(kvm, match);
    kvm_free_assigned_device(kvm, match);

    out:
    mutex_unlock(&kvm->lock);
    return r;
}
```

### 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.
4. Getting Help and Giving Feedback

4.1. Do You Need Help?
If you experience difficulty with a procedure described in this documentation, visit the Red Hat Customer Portal at http://access.redhat.com. Through the customer portal, you can:

» search or browse through a knowledgebase of technical support articles about Red Hat products.
» submit a support case to Red Hat Global Support Services (GSS).
» access other product documentation.

Red Hat also hosts a large number of electronic mailing lists for discussion of Red Hat software and technology. You can find a list of publicly available mailing lists at https://www.redhat.com/mailman/listinfo. Click on the name of any mailing list to subscribe to that list or to access the list archives.

4.2. We Need Feedback!
If you find a typographical error in this manual, or if you have thought of a way to make this manual better, we would love to hear from you! Please submit a report in Bugzilla: http://bugzilla.redhat.com/ against the product Red Hat Storage.

When submitting a bug report, be sure to mention the manual's identifier: doc-Administration_Guide

If you have a suggestion for improving the documentation, try to be as specific as possible when describing it. If you have found an error, please include the section number and some of the surrounding text so we can find it easily.
Part I. Introduction
Chapter 1. Introducing Red Hat Storage

Red Hat Storage is software only, scale-out storage that provides flexible and agile unstructured data storage for the enterprise. Red Hat Storage 2.0 provides new opportunities to unify data storage and infrastructure, increase performance, and improve availability and manageability in order to meet a broader set of an organization's storage challenges and needs.

glusterFS, a key building block of Red Hat Storage, is based on a stackable user space design and can deliver exceptional performance for diverse workloads. glusterFS aggregates various storage servers over network interconnects into one large parallel network file system. The POSIX compatible glusterFS servers, which use XFS file system format to store data on disks, can be accessed using industry standard access protocols including NFS and SMB.

Red Hat Storage can be deployed in the private cloud or datacenter using Red Hat Storage Server for On-premise. Red Hat Storage can be installed on commodity servers and storage hardware resulting in a powerful, massively scalable, and highly available NAS environment. Additionally, Red Hat Storage can be deployed in the public cloud using Red Hat Storage Server for Public Cloud, for example, within the Amazon Web Services (AWS) cloud. It delivers all the features and functionality possible in a private cloud or datacenter to the public cloud by providing massively scalable and high available NAS in the cloud.

Red Hat Storage Server for On-Premise

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

Red Hat Storage Server for Public Cloud

Red Hat Storage Server for Public Cloud packages glusterFS as an Amazon Machine Image (AMI) for deploying scalable NAS in the AWS public cloud. This powerful storage server provides a highly available, scalable, virtualized, and centrally managed pool of storage for Amazon users.
Chapter 2. Red Hat Storage Architecture

This chapter provides an overview of the Red Hat Storage architecture.

At the heart of the Red Hat Storage design is a completely new view of how storage should be architected. The result is a system that has immense scalability, is highly resilient, and offers extraordinary performance.

In a scale-out system, one of the biggest challenges is to keep track of the logical and physical location of data (and metadata). Most distributed systems solve this problem by creating a metadata server which keeps track of data and location of metadata. This creates both a central point of failure and a huge performance bottleneck. As traditional systems add more files, more servers, or more disks, the central metadata server becomes a performance bottleneck. Unlike other traditional solutions, Red Hat Storage does not need a metadata server and locates files algorithmically using the elastic hashing algorithm. This no-metadata server architecture ensures better performance, linear scalability, and reliability.

Figure 2.1. Red Hat Storage Architecture

2.1. Red Hat Storage Server for On-Premise Architecture

The Red Hat Storage Server enables enterprises to treat physical storage as a virtualized, scalable, and centrally managed pool of storage by using commodity storage hardware.

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

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

Figure 2.2. Red Hat Storage Server for On-Premise Architecture

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

2.2. Red Hat Storage Server for Public Cloud

Red Hat Storage Server for Public Cloud packages glusterFS as an Amazon Machine Image (AMI) for deploying scalable NAS in the AWS public cloud. This powerful storage server provides a highly available, scalable, virtualized, and centrally managed pool of storage for Amazon users. The Red Hat Server for Public Cloud provides highly available storage within AWS. Synchronous N-way replication across AWS Availability Zones provides high availability within an AWS Region. Asynchronous Geo-replication provides continuous data replication to ensure high availability across AWS regions. The glusterFS global namespace capability aggregates disk and memory resources into a unified storage volume that is abstracted from the physical hardware.

Red Hat Storage Server for Public Cloud is the only high availability (HA) storage solution available for AWS. It simplifies the task of managing unstructured file data whether you have few terabytes of storage or multiple petabytes. This unique HA solution is enabled by the synchronous file replication capability built into the glusterFS.
Figure 2.3. Red Hat Storage Server for Public Cloud Architecture
Chapter 3. Key Features

This chapter lists the key features of Red Hat Storage 2.0.

3.1. Elasticity

Storage volumes are abstracted from the underlying hardware and can grow, shrink, or be migrated across physical systems as necessary. Storage system servers can be added or removed on-the-fly with data rebalanced across the trusted storage pool. Data is always online and there is no application downtime. File system configuration changes are accepted at runtime and propagated throughout the trusted storage pool allowing changes to be made dynamically as workloads fluctuate or for performance tuning.

3.2. No Metadata with the Elastic Hash Algorithm

Unlike other storage systems with a distributed file system, Red Hat Storage does not create, store, or use a separate index of metadata in any way. Instead, Red Hat Storage places and locates files algorithmically. All storage node servers in the trusted storage pool have the intelligence to locate any piece of data without looking it up in an index or querying another server. Red Hat Storage uses an elastic hashing algorithm to locate data in the storage pool removing the common source of I/O bottlenecks and single point of failure. Data access is fully parallelized and performance scales linearly.

3.3. Scalability

Red Hat Storage is designed to scale for both performance and capacity. This implies that the system should be able to scale up (or down) along multiple dimensions. By aggregating the disk, CPU, and I/O resources of large numbers of commodity hardware — an enterprise should be able to create one very large and performant storage pool. If the enterprise wants to add more capacity to a scale-out system, they can do so by adding more disks. If the enterprise wants to gain performance, they can do so by deploying disks between more server nodes.

3.4. High Availability and Flexibility

Synchronous n-way file replication ensures high data availability and local recovery. Asynchronous geo-replication ensures resilience across datacenters and regions. Both n-way synchronous and Geo-Rep asynchronous data replication are supported in the private cloud, datacenter, public cloud, and hybrid cloud environments. Within the AWS cloud, Red Hat Storage supports n-way synchronous replication across availability zones and Geo-Rep asynchronous replication across AWS Regions. In fact, Red Hat Storage is the only way to ensure high availability for NAS storage within the AWS infrastructure.

3.5. Flexibility

Runs in user space, eliminating the need for complex kernel patches or dependencies. You can also reconfigure storage performance characteristics to meet your changing storage needs.

3.6. No Application Rewrites

The POSIX compatible Red Hat Storage servers, which use XFS file system format to store data on disks, can be accessed using industry standard access protocols including NFS and SMB. So there is no need to rewrite applications when moving data to the cloud as is the case with traditional cloud-based object storage solutions.
3.7. Simple Management

Red Hat Storage lets you build a scale-out storage system that is highly secure within minutes. It provides a very simple, single command for storage management. It also includes performance monitoring and analysis tools like Top and Profile. Top provides visibility into the workload pattern and Profile provides performance statistics over a user-defined time period for metrics including latency and amount of data read or written.

3.8. Modular, Stackable Design

Enterprises can configure and tune Red Hat Storage Servers to deliver high performance for a wide range of workloads. The stackable design allows users to combine modules as needed depending on storage requirements and workload profiles.

3.9. Unified File and Object Storage

Unified File and Object Storage (UFO) unifies the simplicity of NAS storage with the power of 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.

3.10. Hadoop Compatible Storage

Red Hat Storage 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 Red Hat Storage seamlessly. This functionality opens up data within Hadoop deployments to any file-based or object-based application.
Chapter 4. Use Case Examples
This chapter provides use case examples that describe various deployment environments using Red Hat Storage 2.0.

Note
The use cases show general or common situations particular to a specific use of the product. Actual needs and requirements vary from customer to customer and business to business.

4.1. Use Case 1: Using Red Hat Storage for Data Archival
Enterprises, today, face an explosion of data which are driven by varied applications such as virtualization, collaboration, business intelligence, data warehousing, e-mail, ERP/CRM, and media. Data retention requirements increase the problem since retaining multiple versions of data for a prolonged period of time requires more storage.

Red Hat Storage Server for data archival address the challenges associated with rapid archive growth, highly distributed users, and siloed storage pools. Red Hat Storage Server provides an open source, scale-out network attached storage (NAS) and object storage software solution that is designed to work seamlessly with industry standard x86 servers. It also provides freedom of choice to customers by allowing them to deploy cost-effective, scalable and highly available storage without compromising on scale or performance.

Red Hat Storage Server can be deployed on-premise, in private clouds, in public cloud infrastructures or hybrid cloud environments and is optimized for storage intensive enterprise workloads including high-performance computing, nearline archival and rich media content delivery.

Figure 4.1. Use Case 1: Red Hat Storage for Data Archival and Nearline
4.1.1. Key Features of Red Hat Storage Server for Nearline and Archival Use Case

This section describes the key features of Red Hat Storage for nearline and data archival use case.

- **Elastic Scalability**
  Storage volumes are abstracted from the hardware, allowing each of the volume to be managed independently. Volumes can be expanded or shrunk by adding or removing systems from the storage pool, or by adding or removing storage from individual machines in the pool, all while data remains available and with no application interruption.

- **Compatibility**
  Due to native POSIX compatibility and support for SMB, NFS, and HTTP protocols, Red Hat Storage Server is readily supported by industry standard storage management and backup software.

- **High Availability**
  Automatic replication ensures high levels of data protection and resiliency, if hardware fails. Self-healing capabilities restore data to the correct state following recovery.

- **Unified Global Namespace**
  An unified global namespace aggregates disk and memory resources into a single common pool, simplifying management of the storage environment and eliminating data silos. Namespaces can be expanded or shrunk dynamically, with no interruption to client access.

- **Efficient Data Access**
  Red Hat Storage Server provides fast and efficient random access, ensuring speedy data recovery when needed.

4.2. Use Case 2: Using Red Hat Storage for High Performance Computing

Enabling technologies and techniques has become more accessible, and in data-intensive industries such as financial services, energy, life sciences, High Performance Computing (HPC) has gained popularity in recent years. These industries continue to face several challenges as they use HPC to gain an edge in their respective industry.

The key challenges enterprises face are scalability and performance. Red Hat Storage Server family provides an open source, scale-out network attached storage (NAS) and object storage software solution that is designed to work seamlessly with industry standard x86 servers.

Red Hat Storage Server is built on Red Hat Enterprise Linux operating system. It provides freedom of choice to customers by allowing them to deploy cost-effective, scalable and highly-available storage without compromising on scale or performance. It can easily be deployed on-premise, in private clouds, in public cloud infrastructures or hybrid cloud environments and is optimized for high-performance computing workloads that demand high bandwidth and throughput performance.
4.2.1. Key Features of Red Hat Storage Server for High Performance Computing Use Case

This section describes the key features of Red Hat Storage for High Performance Computing use case.

- **Petabyte Scalability**
  Red Hat Storage Server’s fully distributed architecture and advanced file management algorithms allow it to support multi-petabyte repositories with ease.

- **High Performance with no bottleneck**
  Red Hat Storage Server enables quick fast file access by algorithmically spreading files evenly throughout the system, without a centralized metadata server. As the nodes can access storage nodes directly, hot spots, choke points, and other I/O bottlenecks are eliminated. Hence, contention for data is reduced and there is no single point of failure.

- **Elastic Scalability**
  Storage volumes are abstracted from hardware, allowing each to be managed independently. Storage can be added or removed from the storage pools while data continues to be available, with no application interruption. Volumes can be expanded or shrink across machines. It can be migrated within the system to rebalance capacity or add/remove systems on-the-fly, allowing HPC environments to scale seamlessly.

- **Infiniband Support**
  Red Hat Storage Server supports IP over Infiniband (IPoIB). Infiniband as a back-end interconnect for the storage pool is recommended as it provides additional options for maximizing performance. Using RDMA as a mount protocol for its native client is a technology preview feature.

- **Compatibility**
  Due to native POSIX compatibility and support for the SMB, NFS and HTTP protocols, Red Hat Storage Server supports existing applications with no code changes required.
4.3. Use Case 3: Using Red Hat Storage for Content Clouds

Today, smartphones, tablets, and laptops has empowered consumers and enterprise users alike to create and consume multimedia content at an astounding rate, from any place and at any time. The resulting deluge of digital information has created challenges for both traditionally media-intensive industries such as entertainment and internet services, as well as the large number of non-media organizations seeking competitive advantage through content. Media repositories of 100 TB are becoming increasingly common, and multi-petabyte repositories a reality for many organizations, the cost and complexity of building and running media storage systems based on traditional NAS and SAN technologies is overwhelming and often not feasible.

Red Hat Storage Server family provides an open source, scale-out network attached storage (NAS) and object storage software solution that is designed to work seamlessly with industry standard x86 servers. It can easily be deployed on-premise, in private clouds, in public cloud infrastructures or hybrid cloud environments and is optimized for storage intensive enterprise workloads including high-performance computing, nearline archival and rich media content delivery.

Figure 4.3. Use Case 3: Using Red Hat Storage for Content Clouds

4.3.1. Key Features of Red Hat Storage Server for Content Clouds Use Case

This section describes the key features of Red Hat Storage for content clouds use case.

- Elasticity
  Storage volumes are abstracted from hardware, allowing each to be managed independently. Storage can be added or removed from the storage pools while data continues to be available, with no application interruption. Volumes can be expanded or shrink across machines. It can be migrated within the system to rebalance capacity or add/remove systems on-the-fly and scale seamlessly.

- Petabyte Scalability
Red Hat Storage Server’s fully distributed architecture and advanced file management algorithms allow it to support multi-petabyte repositories with ease.

**High Performance**

Red Hat Storage Server enables quick fast file access by algorithmically spreading files evenly throughout the system, without a centralized metadata server. As the nodes can access storage nodes directly, hot spots, choke points, and other I/O bottlenecks are eliminated. Hence, contention for data is reduced and there is no single point of failure.

**Compatibility**

Due to native POSIX compatibility and support for the SMB, NFS and HTTP protocols, Red Hat Storage Server supports existing applications with no code changes required.

**Unified File and Object Access**

Files may be accessed through a simple Web Service REST (Representational State Transfer) interface, enabling easy sharing of files across the Internet, without sacrificing the convenience of loading and managing files through native Unix, Linux, and Windows protocols.

**Reliability**

Replication ensures high levels of data protection and resiliency, even in the event of hardware failure. Self-healing capabilities restore data to the correct state after recovery.
Chapter 5. Storage Concepts

This chapter defines common terms relating to file systems and storage used throughout the Red Hat Storage Administration Guide.

**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/
```

**Block Storage**
Block special files or block devices correspond to devices through which the system moves data in the form of blocks. These device nodes often represent addressable devices such as hard disks, CD-ROM drives, or memory-regions. Red Hat Storage supports XFS file system with extended attributes.

**Cluster**
A trusted pool of linked computers, working together closely thus in many respects forming a single computer. In Red Hat Storage terminology a cluster is called as trusted storage pool.

**Client**
The machine which mounts the volume (this may also be a server)

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

**File System**
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](https://en.wikipedia.org/wiki/File_system)

**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](https://en.wikipedia.org/wiki/Filesystem_in_Userspace)
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 glusterFS 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.

N-way Replication
Local synchronous data replication typically deployed across campus or Amazon Web Services Availability Zones.

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

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 = 1000^5 B = 10^15 B.

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


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. Red Hat Storage exports a fully POSIX compatible 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.
Server
The machine (virtual or bare metal) which hosts the actual file system in which data will be stored.

Scale-Up Storage
 Increases the capacity of the storage device, but only in a single dimension. An example might be adding additional disk capacity to a single computer in a trusted storage pool.

Scale-Out Storage
 Increases the capability of a storage device in multiple dimensions. For example adding a server to a trusted storage pool increases CPU, disk capacity, and throughput for the trusted storage pool.

Subvolume
A brick after being processed by at least one translator.

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

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.

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

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

Volfile
Volfile is a configuration file used by glusterFS process. Volfile will be usually located at /var/lib/glusterd/vols/VOLNAME.

Volume
A volume is a logical collection of bricks. Most of the Red Hat Storage management operations happen on the volume.
Part II. Red Hat Storage Administration On-Premise
Chapter 6. Managing the glusterd Service

After installing Red Hat Storage, glusterd service will be started automatically on all the servers in your trusted storage pool. You can also manually start and stop glusterd service manually.

Red Hat Storage allows you to dynamically change the configuration of Red Hat Storage volumes without having to restart servers or remount Red Hat Storage volumes on clients. You can perform this type of elastic volume management using the glusterFS daemon called glusterd.

Using the glusterd command line, logical storage volumes are decoupled from physical hardware, allowing you to grow, shrink and migrate storage volumes without any application downtime. As storage is added, Red Hat Storage volumes are rebalanced across the trusted storage pool making it always available online regardless of changes to the underlying hardware.

6.1. Starting and Stopping glusterd Manually

This section describes how to start and stop glusterd service 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
  ```
Chapter 7. Setting up Trusted Storage Pools

Before you can configure a Red Hat Storage volume, you must create a trusted storage pool consisting of the storage servers that provide 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 trusted.

**Note**

You must not self-probe the first server/localhost. Probing the first server/localhost displays an error message as it is part of the trusted storage pool.

The `glusterd` service must be running on all storage servers that you want to add to the storage pool. See Chapter 6, Managing the `glusterd` Service for more information.

7.1. 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.
   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 all servers using the following command:

   ```
   # gluster peer status
   Number of Peers: 3
   
   Hostname: server2
   Uuid: 5e987bda-16dd-43c2-835b-08b7d55e94e5
   State: Peer in Cluster (Connected)
   
   Hostname: server3
   Uuid: 1e0ca3aa-9ef7-4f66-8f15-cbc348f29ff7
   State: Peer in Cluster (Connected)
   
   Hostname: server4
   Uuid: 3e0caba-9df7-4f66-8e5d-cbc348f29ff7
   State: Peer in Cluster (Connected)
   ```
7.2. Removing Servers from the Trusted Storage Pool

To remove a server from the storage pool:

```bash
# gluster peer detach server
```

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

```bash
# gluster peer detach server4
Detach successful
```
Chapter 8. Setting up Red Hat Storage Volumes

A Red Hat Storage volume is a logical collection of bricks where each brick is an export directory on a server in the trusted storage pool. Most of the Red Hat Storage Server management operations are performed on the volume.

Warning

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

To create a new Red Hat Storage volume in your storage environment, specify the bricks that comprise the Red Hat Storage volume. After you have created a new Red Hat Storage 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 across bricks in the volume. You can use distributed volumes where the requirement is to scale storage and redundancy is either not important or is provided by other hardware/software layers. For more information, see Section 8.3, "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 8.4, "Creating Replicated 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 8.5, "Creating Distributed Replicated Volumes".

Important

Striped, Striped-Replicated, Distributed-Striped, and Distributed-Striped-Replicated volume types are under technology preview. Technology Preview features are not fully supported under Red Hat subscription level agreements (SLAs), may not be functionally complete, and are not intended for production use. However, these features provide early access to upcoming product innovations, enabling customers to test functionality and provide feedback during the development process.

- 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 8.6, "Creating Striped Volumes".
- Striped Replicated – Striped replicated volumes stripes data across replicated bricks in the trusted storage pool. 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 8.7, "Creating Striped Replicated Volumes".
- Distributed Striped - Distributed striped volumes stripe data across two or more nodes in the trusted storage pool. You should use distributed striped volumes where the requirement is to scale storage and in high concurrency environments where accessing very large files is critical. For more information, see Section 8.8, "Creating Distributed Striped Volumes".
- Distributed Striped Replicated – Distributed striped replicated volumes distributes striped data across replicated bricks in the trusted storage pool. 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. Configuration of this volume type is supported only for Map Reduce workloads. For more information, see Section 8.9, “Creating Distributed Striped Replicated Volumes”.

**Note**

Red Hat Storage supports IP over Infiniband (IPoIB). You must install Infiniband packages on all Red Hat Storage servers and clients. Run the following command to install Infiniband packages:

```
# yum groupinstall "Infiniband Support"
```

Red Hat Storage support for RDMA over Infiniband is a technology preview feature.

### 8.1. Formatting and Mounting Bricks

Red Hat supports only formatting a Logical Volume by using the XFS file system on the bricks with few modifications to improve performance. Ensure to format bricks using XFS on Logical Volume Manager before adding it to a Red Hat Storage volume. Red Hat Storage uses extended attributes on files, so you must increase the inode size to 512 bytes from the default 256 bytes by running the following command:

```
# mkfs.xfs -i size=512 DEVICE
```

You can now mount the bricks on the Red Hat Storage servers.

To mount bricks:

1. Obtain the UUID (universally unique identifier) of the device using the following command:

   ```
   # blkid DEVICE
   ```

2. Create a directory to link the brick using the following command:

   ```
   # mkdir /mountpoint
   ```

3. Add an entry to `/etc/fstab` using the obtained UUID from the `blkid` command:

   ```
   UUID=uuid /mountpoint xfs defaults 1 2
   ```

4. Mount the brick using the following command:

   ```
   # mount /mountpoint
   ```

5. Run `df -h` command to verify that the brick is successfully mounted:

   ```
   # df -h
   /dev/vg_bricks/lv_exp1 16G 1.2G 15G 7% /exp1
   ```
Important

While creating a logical volume, Red Hat recommends that you must allocate 15% - 20% of free space to take advantage of Red Hat Storage volume Snapshotting feature, which will be available in a future release of Red Hat Storage. If your brick is created by formatting a raw disk partition with XFS file system, Red Hat does not support this configuration.

If you reuse a brick of a deleted volume while creating a new volume, the volume creation fails as the brick contains volume ID of its previously associated volume. You must ensure to delete the extended attributes set on the brick directory before reusing the brick by executing the following commands:

```
# setfattr -x trusted.glusterfs.volume-id brick
# setfattr -x trusted.gfid brick
```

8.2. Encrypted Disk

Red Hat Storage provides the ability to create bricks on encrypted devices so that access to data is restricted. You can create bricks on encrypted disk and use them to create Red Hat Storage volumes.

For information on creating encrypted disk, refer to Appendix C, Disk Encryption of Red Hat Enterprise Linux 6 Installation Guide.

8.3. Creating Distributed Volumes

In a distributed volume files are spread 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.

Warning

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.
To create a distributed volume

1. Create a trusted storage pool as described earlier in Section 7.1, “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 two storage servers using tcp:

   ```
   # gluster volume create test-volume server1:/exp1 server2:/exp2
   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: 2
Transport-type: tcp
Bricks:
Brick1: server1:/exp1
Brick2: server2:/exp2
```

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 10.1, “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 8.10, “Starting Volumes” for details.

8.4. Creating Replicated Volumes

Important
Creating replicated volume with rep_count > 2 is under technology preview. Technology Preview features are not fully supported under Red Hat subscription level agreements (SLAs), may not be functionally complete, and are not intended for production use. However, these features provide early access to upcoming product innovations, enabling customers to test functionality and provide feedback during the development process. As Red Hat considers making future iterations of Technology Preview features generally available, we will provide commercially reasonable efforts to resolve any reported issues that customers experience when using these features.

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 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.
To create a replicated volume

1. Create a trusted storage pool as described earlier in Section 7.1, “Adding Servers to Trusted Storage Pool”.
2. Create the replicated volume:

```bash
# 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:

```bash
# 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.
```

The order in which the bricks are specified determines the mirroring of bricks with each other. For example, first \( n \) bricks, where \( n \) is the replica count. Hence, the first two bricks specified will mirror each other, and the third and fourth bricks will mirror each other.

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 10.1, “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 8.10, “Starting Volumes” for details.

8.5. Creating Distributed Replicated Volumes
**Important**

Creating distributed-replicated volume with rep_count > 2 is under technology preview. Technology Preview features are not fully supported under Red Hat subscription level agreements (SLAs), may not be functionally complete, and are not intended for production use. However, these features provide early access to upcoming product innovations, enabling customers to test functionality and provide feedback during the development process. As Red Hat considers making future iterations of Technology Preview features generally available, we will provide commercially reasonable efforts to resolve any reported issues that customers experience when using these features.

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 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 8.3. Illustration of a Distributed Replicated Volume
To create a distributed replicated volume

1. Create a trusted storage pool as described earlier in Section 7.1, “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.
   ```

   The order in which the bricks are specified determines the mirroring of bricks with each other. For example, first \( n \) bricks, where \( n \) is the replica count. Hence, the first two bricks specified will mirror each other, and the third and fourth bricks will mirror each other.

   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 10.1, “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 8.10, “Starting Volumes” for details.

### 8.6. Creating Striped Volumes

**Important**

Striped volume is a technology preview feature. Technology Preview features are not fully supported under Red Hat subscription level agreements (SLAs), may not be functionally complete, and are not intended for production use. However, these features provide early access to upcoming product innovations, enabling customers to test functionality and provide feedback during the development process.

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

Figure 8.4. Illustration of a Striped Volume

To create a striped volume

1. Create a trusted storage pool as described earlier in Section 7.1, “Adding Servers to Trusted Storage Pool”.
2. Create the striped volume:

   ```bash
   # 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:

   ```bash
   # 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 10.1, “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 8.10, “Starting Volumes” for details.
8.7. Creating Striped Replicated Volumes

**Important**

Striped-Replicated volume is a technology preview feature. Technology Preview features are not fully supported under Red Hat subscription level agreements (SLAs), may not be functionally complete, and are not intended for production use. However, these features provide early access to upcoming product innovations, enabling customers to test functionality and provide feedback during the development process.

**Note**

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

Striped replicated volumes stripes data across replicated bricks in the trusted storage pool. 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.

![Figure 8.5. Illustration of a Striped Replicated Volume](image)

**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 7.1, “Adding Servers to Trusted Storage Pool”.

2. Create a striped replicated volume:

```bash
# 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:

```bash
# gluster volume create test-volume stripe 2 replica 2 transport tcp server1:/exp1 server2:/exp3 server3:/exp2 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:

```bash
# 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.
```

The order in which the bricks are specified determines the mirroring of bricks with each other. For example, first \( n \) bricks, where \( n \) is the replica count. Hence, the first two bricks specified will mirror each other, and the third and fourth bricks will mirror each other.

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 10.1, “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 8.10, “Starting Volumes” for details.

### 8.8. Creating Distributed Striped Volumes

**Important**

Distributed-Striped volume is a technology preview feature. Technology Preview features are not fully supported under Red Hat subscription level agreements (SLAs), may not be functionally complete, and are not intended for production use. However, these features provide early access to upcoming product innovations, enabling customers to test functionality and provide feedback during the development process.

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

**Note**

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

1. Create a trusted storage pool as described earlier in Section 7.1, “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 four storage servers:

   ```
   # gluster volume create test-volume stripe 2 transport tcp server1:/exp1 server1:/exp2 server2:/exp3 server2:/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 10.1, “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 8.10, “Starting Volumes” for details.
Important

Distributed-Striped-Replicated volume is a technology preview feature. Technology Preview features are not fully supported under Red Hat subscription level agreements (SLAs), may not be functionally complete, and are not intended for production use. However, these features provide early access to upcoming product innovations, enabling customers to test functionality and provide feedback during the development process.

Distributed striped replicated volumes distributes striped data across replicated bricks in the trusted storage pool. 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.

Figure 8.7. Illustration of a Distributed Striped Replicated Volume

To create a distributed striped replicated volume
1. Create a trusted storage pool as described earlier in Section 7.1, “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 server1:/exp2 server2:/exp3 server2:/exp4 server3:/exp5 server3:/exp6 server4:/exp7 server4:/exp8
   Creation of test-volume has been successful
   Please start the volume to access data.
   ```

   The order in which the bricks are specified determines the mirroring of bricks with each other. For example, first \( n \) bricks, where \( n \) is the replica count. Hence, the first two bricks specified will mirror each other, and the third and fourth bricks will mirror each other.

   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 10.1, “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 8.10, “Starting Volumes” for details.

8.10. 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
  ```
Chapter 9. Accessing Data - Setting Up Clients

You can access Red Hat Storage volumes in multiple ways. You can use Native Client method for high concurrency, performance and transparent failover in GNU/Linux clients. You can also use NFS v3 to access Red Hat Storage volumes. Linux and other operating systems that support the NFSv3 standard may use NFS to access the Red Hat Storage volumes. However, there may be some differences in implementation of the NFSv3 standard by different operating systems which may lead to some issues.

You can contact your Red Hat representative for more information on compatibility of Red Hat Storage Server for your specific client operating system and known issues that may exist.

You can use SMB (Server Message Block) to access Red Hat Storage volumes when using Microsoft Windows as well as Samba clients. For this access method, Samba packages need to be present on the client side.

9.1. Native Client

Native Client is a FUSE-based client running in user space. Native Client is the recommended method for accessing Red Hat Storage volumes when high concurrency and high write performance is required.

This section introduces Native Client and explains how to install the software on client machines. This section also describes how to mount Red Hat Storage volumes on clients (both manually and automatically) and how to verify that the Red Hat Storage volume has mounted successfully.

9.1.1. Installing Native Client

After you have successfully installed your client operating system, you must first register the target system to Red Hat Network and subscribe to the Red Hat Enterprise Linux Server channel.

To subscribe to the Red Hat Enterprise Linux Server channel using RHN Classic:

1. Run the `rhn_register` command to register the system with Red Hat Network. To complete registration successfully you will need to supply your Red Hat Network username and password.
   # rhn_register

   In the select operating system release page, select All available updates and follow the on screen prompts and complete the registration of the system.
   The system is now registered to rhel-x86_64-server-6 channel.

2. **Subscribe to Red Hat Storage Native Client**
   You must subscribe the system to the Red Hat Storage Native Client channel using either the web interface to Red Hat Network or the command line `rhn-channel` command.
   A. Using the `rhn-channel` command
      Run the `rhn-channel` command to subscribe the system to Red Hat Storage Native Client channel. The command which need to be run is:
      ```bash
      # rhn-channel --add --channel=rhel-x86_64-server-rhsclient-6
      ```
   B. **Using the Web Interface to Red Hat Network.**
      To add a channel subscription to a system from the web interface:
      i. Log on to Red Hat Network (http://rhn.redhat.com).
      ii. Move the mouse cursor over the Subscriptions link at the top of the screen, and then click the Registered Systems link in the menu that appears.
iii. Select the system to which you are adding **Red Hat Storage Native Client** channel from the list presented on the screen, by clicking the name of the system.

iv. Click **Alter Channel Subscriptions** in the **Subscribed Channels** section of the screen.

v. On this screen, expand the node for Additional Services Channels for **Red Hat Enterprise Linux 6 for x86_64** for RHEL 6 or Additional Services Channels for **Red Hat Enterprise Linux 5 for x86_64** for RHEL 5.

vi. Click the **Change Subscriptions** button to finalize the changes.

After the page refreshes, select the **Details** tab to verify if your system is subscribed to the appropriate channels.

Run the following command to verify if the system is registered successfully.

```bash
# rhn-channel -l
rhel-x86_64-server-6
rhel-x86_64-server-rhsclient-6
```

The system is now registered with Red Hat Network and subscribed to the **Red Hat Storage Native Client** channel. Now install the native client RPMs using the following command:

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

---

**Important**

All the clients must be of same version. Red Hat strongly recommends you to upgrade your clients before you upgrade the server.

---

If you are using RHEL 5.x machines, you must load FUSE modules before mounting the Red Hat Storage volumes. Execute the following command to load FUSE modules:

```bash
# modprobe fuse
```

For more information on loading modules at boot time, see [https://access.redhat.com/knowledge/solutions/47028](https://access.redhat.com/knowledge/solutions/47028).

---

### 9.1.2. Mounting Red Hat Storage Volumes

After installing Native Client, you must mount Red Hat Storage volumes to access data. There are two methods you can choose:

- Section 9.1.2.1, “Manually Mounting Volumes”
- Section 9.1.2.2, “Automatically Mounting Volumes”

After mounting a volume, you can test the mounted volume using the procedure described in Section 9.1.2.3, “Testing Mounted Volumes”.
Server names selected during creation of volumes should be resolvable in the client machine. You can use appropriate /etc/hosts entries or a DNS server to resolve server names to IP addresses.

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` - name of the backup volfile server to mount the client. If this 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.
- `fetch-attempts=number` - number of attempts to fetch volume files while mounting a volume. This option is useful when you mount a server with multiple IP addresses or when round-robin DNS is configured for the server name.
- `log-level` - logs only specified level or higher severity messages in the log-file.
- `log-file` - logs the messages in the specified file.
- `direct-io-mode=[enable|disable]`
- `ro` - mounts the file system as read only.
- `acl` - enables POSIX Access Control List on mount.
- `selinux` - enables handling of SELinux xattrs through the mount point.
- `background-qlen=length` - this option enables FUSE to handle n number of requests to be queued before stopping to accept new requests. Default value of n is 64.
- `enable-ino32` - this option enables file system to present 32-bit inodes instead of 64-bit inodes.

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
```

9.1.2.1. Manually Mounting Volumes

To manually mount a Red Hat Storage 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
```
9.1.2.2. Automatically Mounting Volumes

You can configure your system to automatically mount the Red Hat Storage volume each time your system starts.

The server specified in the mount command is only used to fetch the glusterFS 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).

To automatically mount a Red Hat Storage 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
```

9.1.2.3. Testing Mounted Volumes

To test mounted volumes

Use the following command:

```
# mount
```

For example, if the Red Hat Storage volume was successfully mounted, the output of the mount command on the client will display an entry like the following:

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

Use the following command:

```
# df
```

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
```

## 9.2. NFS

You can use NFS v3 to access to Red Hat Storage volumes. Linux and other operating systems that support the NFSv3 standard may use NFS to access the Red Hat Storage volumes. However, there may be some differences in implementation of the NFSv3 standard by different operating systems which may lead to some issues. You can contact your Red Hat representative for more information on compatibility of Red Hat Storage Server for your specific client operating system and known issues that may exist.

Red Hat Storage 2.0 includes network lock manager (NLM) v4. NLM protocol allows NFSv3 clients to lock files across the network. NLM is required to make applications running on top of NFSv3 mount points to use the standard fcntl() (POSIX) and flock() (BSD) lock system calls to synchronize access across clients.

This section describes how to use NFS to mount Red Hat Storage volumes (both manually and automatically) and how to verify that the volume has been mounted successfully.

### 9.2.1. Using NFS to Mount Red Hat Storage Volumes

You can use either of the following methods to mount Red Hat Storage volumes:

- Section 9.2.1.1, "Manually Mounting Volumes Using NFS"
- Section 9.2.1.2, "Automatically Mounting Volumes Using NFS"

After mounting a volume, you can test the mounted volume using the procedure described in Section 9.2.1.3, "Testing Volumes Mounted Using NFS".

#### 9.2.1.1. Manually Mounting Volumes Using NFS

To manually mount a Red Hat Storage 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**

glusterFS NFS server does not support UDP. If a NFS client such as Solaris client, connects by default using UDP, the following message appears:

 requested NFS version or transport protocol is not supported.
To connect using TCP
» Add the following option to the mount command:

```bash
-o mountproto=tcp
```

For example:

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

To mount Red Hat Storage NFS server from a Solaris Client
» Use the following command:

```bash
# mount -o proto=tcp,vers=3 nfs://hostname-or-IPaddress:38467/volname mountdir
```

For example:

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

9.2.1.2. Automatically Mounting Volumes Using NFS
You can configure your system to automatically mount Red Hat Storage volumes using NFS each time the system starts.

To automatically mount a Red Hat Storage 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
```

Note
glusterFS NFS server does not support UDP. If a NFS client such as Solaris client, connects by default using UDP, the following message appears:

`requested NFS version or transport protocol is not supported.`

To connect using TCP
» Add the following entry in `/etc/fstab` file:

```
hostname-or-IPaddress:/volname mountdir nfs defaults,_netdev,mountproto=tcp 0 0
```

For example:

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

To automount NFS mounts
Red Hat Storage supports Linux, UNIX, and similar operating system's 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.

9.2.1.3. Testing Volumes Mounted Using NFS
You can confirm that Red Hat Storage directories are mounting successfully.

**To test mounted volumes**

- Use the mount command by entering the following:

  ```
  # mount
  ```

  For example, the output of the mount command on the client will display an entry similar to the following:

  ```
  server1:/test-volume on /mnt/glusterfs type nfs (rw,vers=3,addr=server1)
  ```

- Use the `df` command by entering the following:

  ```
  # df
  ```

  For example, the output of `df` command on the client will display the aggregated storage space from all the bricks in a volume.

  ```
  # df -h /mnt/glusterfs
  Filesystem              Size   Used   Avail %Used 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
  ```

9.2.2. Troubleshooting NFS
This section describes the most common troubleshooting issues related to NFS.

9.2.2.1. `mount` command on NFS client fails with “RPC Error: Program not registered”
Start rpcbind service on the NFS server.

This error is encountered when the server has not started correctly.

Start the rpcbind service by running the following command:

```
# /etc/init.d/rpcbind start
```
9.2.2.2. 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:

```
```

1. Start the rpcbind service on the NFS server by running the following command:

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

After starting rpcbind service, glusterFS NFS server needs to be restarted.

2. Stop another NFS server running on the same machine.

   Such an error is also seen when there is another NFS server running on the same machine but it is not the glusterFS NFS server. On Linux systems, this could be the kernel NFS server. Resolution involves stopping the other NFS server or not running the glusterFS NFS server on the machine. Before stopping the kernel NFS server, ensure that no critical service depends on access to that NFS server’s exports.

   On Linux, kernel NFS servers can be stopped by using either of the following commands depending on the distribution in use:

```
# /etc/init.d/nfs-kernel-server stop
# /etc/init.d/nfs stop
```


9.2.2.3. NFS server start-up fails with “Port is already in use” error in the log file.

Another glusterFS NFS server is running on the same machine.

This error can arise in case there is already a glusterFS NFS server running on the same machine. This situation can be confirmed from the log file, if the following error lines exist:
To resolve this error one of the glusterFS NFS servers will have to be shutdown. At this time, glusterFS NFS server does not support running multiple NFS servers on the same machine.

### 9.2.2.4. mount command takes too long to finish.

Start `rpcbind` service on the NFS client.

The problem is that the `rpcbind` service is not running on the NFS client. The resolution for this is to start `rpcbind` service by running the following command:

```bash
# /etc/init.d/rpcbind start
```

### 9.2.2.5. mount command fails with NFS server failed error.

Mount command fails with following error

```bash
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 respond 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:

   ```bash
   option rpc-auth.addr.namelookup off
   ```

   **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 by default.
glusterFS NFS server supports version 3 of NFS protocol by default. In recent Linux kernels, the default NFS version has been changed from 3 to 4. It is possible that the client machine is unable to connect to the glusterFS NFS server because it is using version 4 messages which are not understood by glusterFS NFS server. The timeout can be resolved by forcing the NFS client to use version 3. The **vers** option to `mount` command is used for this purpose:

```
# mount nfsserver:export -o vers=3 mount-point
```

### 9.2.2.6. showmount fails with `clnt_create: RPC: Unable to receive`

Check your firewall setting to open ports 111 for portmap requests/replies and glusterFS NFS server requests/replies. glusterFS NFS server operates over the following port numbers: 38465, 38466, and 38467.

### 9.2.2.7. 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 glusterFS NFS 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 NFS 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
```

### 9.3. SMB

You can use SMB (Server Message Block) protocol to access to Red Hat Storage volumes when using Microsoft Windows as well as Linux clients. SMB is also known as CIFS (Common Internet File System). For this access method, SMB Client (CIFS) packages need to be present on the client side. You can export the glusterFS mount point as the Samba export on the server, and then mount it using the SMB protocol on the client.

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

**Note**

SMB access using the Mac OS X Finder is not supported. You can, however, use the Mac OS X command line to access Red Hat Storage volumes using SMB.

If you are using Samba with **Windows Active Directory**, you must install the following packages on the linux client using the following commands:
# yum install samba-winbind
# yum install samba-client
# yum install krb5-workstation

## 9.3.1. Mounting Red Hat Storage Volumes as SMB Shares

You can use either of the following methods to mount Red Hat Storage volumes using SMB:

- [Section 9.3.1.3, “Manually Mounting Volumes Using SMB on Red Hat Enterprise Linux and Windows”](#)
- [Section 9.3.1.4, “Automatically Mounting Volumes Using SMB on Red Hat Enterprise Linux and Windows”](#)

After mounting a volume, you can test the mounted volume using the procedure described in [Section 9.3.1.5, “Testing Volumes Mounted Using SMB on Red Hat Enterprise Linux and Windows”](#).

### 9.3.1.1. Exporting Red Hat Storage Volumes Through Samba

You can use Samba to export Red Hat Storage volumes through the SMB protocol.

**To export volumes through SMB protocol**

1. Mount a Red Hat Storage volume. For more information on mounting volumes, see [Section 9.1.2, “Mounting Red Hat Storage Volumes”](#).
2. Setup the Samba configuration to export the mount point of the Red Hat Storage volume.
   - For example, if a Red Hat Storage volume is mounted on `/mnt/gluster`, you must edit the `/etc/samba/smb.conf` file to enable sharing of Red Hat Storage volumes over SMB. Open the `/etc/samba/smb.conf` file in a text editor and add the following lines for a simple configuration:

   ```
   [glustertest]
   comment = For testing a Red Hat Storage volume exported over SMB
   path = /mnt/gluster
   read only = no
   guest ok = yes
   ```

   Save the changes and start or restart the SMB (Server Message Block) service using your system's init scripts (`/etc/init.d/smb` [re]start).
3. Set the SMB password using the following command:

   ```
   # smbpasswd -a username
   ```

   You will be prompted for a password, provide the SMB password. This password will be used during the SMB mount.

   **Note**

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

### 9.3.1.2. Automatically Exporting Red Hat Storage Volumes Through Samba

When you start a volume using `gluster volume start VOLNAME` command, the volume is automatically exported through Samba on all Red Hat Storage servers running Samba. The Red Hat
Storage volume is mounted using the Red Hat Storage Native Client at /mnt/samba/VOLNAME. It is exported as a Samba share named as **gluster-VOLNAME**.

To disable automatic mounting of volumes through Samba, rename the **S30samba-start.sh** located at /var/lib/glusterd/hooks/1/start/post to **K30samba-start.sh**.

For more information on the above scripts, see Section 14.2, “Prepackaged Scripts”.

### 9.3.1.3. Manually Mounting Volumes Using SMB on Red Hat Enterprise Linux and Windows

You can manually mount Red Hat Storage volumes using SMB on Red Hat Enterprise Linux and Microsoft Windows-based client machines.

**To manually mount a Red Hat Storage volume using SMB on Red Hat Enterprise Linux**

» Mount the Samba exported SMB share using the following command:

```
# mount -t cifs Samba_Server_IP_Address:/Share_Name Mount_Point
```

For example, if a Red Hat Storage volume is exported through SMB using the /etc/samba/smb.conf file with the following entry:

```
[glustertest]
comment = For testing a Red Hat Storage volume exported over SMB
path = /mnt/gluster
read only = no
guest ok = yes
```

Perform the SMB mount of the Red Hat Storage volume using the following command:

```
# mount -t cifs 192.168.1.60:/glustertest /mnt/smb
```

**To manually mount a Red Hat Storage volume using SMB on Windows**

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. In the **Folder** text box, enter the path of the server and the shared resource in the following format: `\\SERVER_NAME\VOLNAME`.
4. Click **Finish**.

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

Alternatively, to manually mount a Red Hat Storage volume using SMB

1. Click **Start**, and then click **Run**.
2. In the **Open** box, enter **cmd**.
3. Enter `net use z: `\\SERVER_NAME\VOLNAME`, where `z:` is the drive letter you want to assign to the shared volume.
   For example, `net use y: `\\server1\test-volume``

### 9.3.1.4. Automatically Mounting Volumes Using SMB on Red Hat Enterprise Linux and Windows

You can configure your system to automatically mount Red Hat Storage volumes using SMB on Microsoft
Windows-based clients each time the system starts.

**To automatically mount a Red Hat Storage volume using SMB on Red Hat Enterprise Linux**

- To automatically mount a volume, edit the `/etc/fstab` file and add the following line:
  
  ```
  hostname-or-IPaddress:/Share_Name mountdir smb credentials=filename, _netdev 0 0
  ```

  For example,
  
  ```
  server1:/glustertest /mnt/glusterfs smb credentials=/etc/samba/passwd, _netdev 0 0
  ```

  You must specify the filename and its path that contains username and/or password in the `credentials` option in `/etc/fstab` file. Refer to the `mount.cifs` man page for more information.

**To automatically mount a Red Hat Storage volume using SMB on Windows**

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. In the **Folder** text box, enter the path of the server and the shared resource in the following format: `\SERVER_NAME\VOLNAME`.
4. Click the **Reconnect at logon** checkbox.
5. Click **Finish**.

**9.3.1.5. Testing Volumes Mounted Using SMB on Red Hat Enterprise Linux and Windows**

You can confirm that Red Hat Storage directories are mounting successfully.

**To test mounted volumes on Red Hat Enterprise Linux**

- Use the `smbstatus` command by entering the following:

  ```
  # smbstatus -S
  ```

  For example, the output of the status command on the client will display an entry similar to the following:

  ```
  Service    pid    machine             Connected at
  ________________________________
  glustertest 11967   __ffff_192.168.1.60  Mon Aug 6 02:23:25 2012
  ```

**To test mounted volumes on Windows**

- You can confirm that Red Hat Storage directories are mounting successfully by navigating to the directory using Windows Explorer.

**9.4. Configuring Automated IP Failover for NFS and SMB**

In replicated volume environment, you can configure Cluster Trivial Database (CTDB) to provide high availability for NFS and SMB exports. CTDB adds virtual IP addresses (VIPS) and a heartbeat service to...
Red Hat Storage Server.

When a node in the trusted storage pool fails, CTDB enables a different node to take over the IP address of the failed node. This ensures the IP addresses for the services provided are always available.

**Note**
EC2 (Amazon Elastic Compute Cloud) does not support VIPs, hence CTDB is not supported with Red Hat Storage for Amazon Web Services.

### 9.4.1. Setting Up CTDB

Perform the following to setup CTDB:

1. Create a replicated volume.
   Ensure that the bricks are in different machines.

2. Update the `META=all` to the newly created volume name on all Red Hat Storage servers which require IP failover in the hook scripts available at
   `/var/lib/glusterd/hooks/1/start/post/S29CTDBsetup.sh` and
   `/var/lib/glusterd/hooks/1/stop/pre/S29CTDB-teardown.sh`.

3. Start the volume.
   The `S29CTDBsetup.sh` script runs on all Red Hat Storage servers and adds the following lines to the `[global]` section of your Samba configuration
   ```
   clustering = yes
   idmap backend = tdb2
   ```

   The script stops Samba server, modifies Samba configuration, adds an entry in `/etc/fstab/` for the mount, and mounts the volume at `/gluster/lock`. It also enables automatic start of CTDB service on a reboot.

**Note**
When you stop a volume, `S29CTDB-teardown.sh` script runs on all Red Hat Storage servers and removes the following lines from `[global]` section of your Samba configuration

   ```
   clustering = yes
   idmap backend = tdb2
   ```

   It also removes an entry in `/etc/fstab/` for the mount and unmount the volume at `/gluster/lock`.

4. Create `/gluster/lock/ctdb` file and add the following entries:

   ```
   CTDB_RECOVERY_LOCK=/gluster/lock/lockfile
   #SMB only
   CTDB_PUBLIC_ADDRESSES=/etc/ctdb/public_addresses
   CTDB_MANAGES_SAMBA=yes #SMB only
   CTDB_NODES=/etc/ctdb/nodes
   ```
5. Create `/gluster/lock/nodes` file and list the IPs of Red Hat Storage servers which require IP failover.

   192.168.1.60
   192.168.1.61
   192.168.1.62
   192.168.1.63

6. Create `/etc/ctdb/public_addresses` file on all Red Hat Storage servers which require IP failover and list the Virtual IPs that CTDB should create. Replace `eth0` with the interface available on that node for CTDB to use.

   192.168.1.20/24 eth0
   192.168.1.21/24 eth0

7. Run the following commands on all Red Hat Storage servers which require IP failover to create symbolic links:

   ```
   # ln -s /gluster/lock/ctdb /etc/sysconfig/ctdb
   # ln -s /gluster/lock/nodes /etc/ctdb/nodes
   ```

   **Note**
   You must ensure to open Port 4379 between the Red Hat Storage servers. In CTDB based high availability environment of NFS and SMB, the locks will not be migrated on failover.

9.4.2. Starting and Verifying your Configuration

Perform the following to start and verify your configuration:

1. Start CTDB service using the following command:

   ```
   # service ctdb start
   ```

2. When CTDB starts it will start Samba automatically. To avoid Samba from starting automatically on reboot, run the following command:

   ```
   # chkconfig smb off
   ```

3. Verify that CTDB is running using the following command:

   ```
   # ctdb status
   # ctdb ip
   # ctdb ping -n all
   ```

4. Mount a Red Hat Storage volume using any one of the VIPs.

   When you shutdown the Red Hat Storage Server serving the VIP (run `ctdb ip` command to find the physical server serving the VIP), there will be a pause for few seconds and then I/O will resume.
9.5. 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. He does not allow anyone in the group to access the file, except for another user, Antony (even if there are other users who 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.

9.5.1. 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 ACL, it inherits the permissions of the default ACLs of the directory.

For a file, ACLs can be configured:

- Per user
- Per group
- Via the effective right mask
- For users not in the user group for the file

9.5.1.1. 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.

<table>
<thead>
<tr>
<th>ACL Entry</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>u:uid:&lt;permission&gt;</td>
<td>Sets the access ACLs for a user. You can specify user name or UID</td>
</tr>
<tr>
<td>g:gid:&lt;permission&gt;</td>
<td>Sets the access ACLs for a group. You can specify group name or GID.</td>
</tr>
<tr>
<td>m:&lt;permission&gt;</td>
<td>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.</td>
</tr>
<tr>
<td>o:&lt;permission&gt;</td>
<td>Sets the access ACLs for users other than the ones in the group for the file.</td>
</tr>
</tbody>
</table>

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
```

### 9.5.1.2. Setting Default ACLs

New files and directories inherit ACL information from their parent directory if that parent has an ACL that contains default entries. You can set default ACL entries only on directories.

**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 ACL set for an individual file can override the default ACL 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.

### 9.5.2. 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
```

### 9.5.3. 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
```

### 9.5.4. Samba and ACLs

If you are using Samba to access Red Hat Storage 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.

### 9.5.5. NFS and ACLs

Red Hat Storage does not support ACLs configuration through NFS, which means `setfacl` and `getfacl` commands does not work. However, ACLs permissions set using Red Hat Storage Native Client is applied on NFS mounts.
Chapter 10. Managing Red Hat Storage Volumes

This chapter describes how to perform common volume management operations on the Red Hat Storage volumes.

10.1. Tuning Volume Options

You can tune volume options, as needed, while the trusted storage pool is online and available.

To tune volume options

- Tune volume options using the following command:

  ```shell
  # gluster volume set VOLNAME OPTION PARAMETER
  ```

  For example, to specify the performance cache size for test-volume:

  ```shell
  # 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:

<table>
<thead>
<tr>
<th>Option</th>
<th>Value Description</th>
<th>Allowed Values</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>auth.allow</td>
<td>IP addresses or hostnames of the clients which should be allowed to access the volume.</td>
<td>Valid hostnames or IP addresses which includes wild card patterns including <em>, such as 192.168.1.</em>, A list of comma separated addresses is accepted, but a single hostname must not exceed 256 characters.</td>
<td>* (allow all)</td>
</tr>
<tr>
<td>auth.reject</td>
<td>IP addresses or hostnames of the clients which should be denied access to the volume.</td>
<td>Valid hostnames or IP addresses which includes wild card patterns including <em>, such as 192.168.1.</em>, A list of comma separated addresses is accepted, but a single hostname must not exceed 256 characters.</td>
<td>none (reject none)</td>
</tr>
<tr>
<td>cluster.min-free-disk</td>
<td>Specifies the percentage of disk space that must be</td>
<td>Percentage of required minimum free disk space</td>
<td>10%</td>
</tr>
<tr>
<td>Feature</td>
<td>Description</td>
<td>Default</td>
<td>Options</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>----------------------------------------------------------------------------------------------</td>
<td>---------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td><code>cluster.self-heal-daemon</code></td>
<td>Allows you to turn-off proactive self-heal on replicated volumes.</td>
<td>on</td>
<td>on/off</td>
</tr>
<tr>
<td><code>cluster.server-quorum-type</code></td>
<td>If set to <code>server</code>, enables the specified volume to participate in quorum. For more information on configuring quorum, see <a href="#">Section 10.9, “Configuring Server-Side Quorum”</a>.</td>
<td>none</td>
<td>none/server</td>
</tr>
<tr>
<td><code>cluster.server-quorum-ratio</code></td>
<td>Sets the quorum percentage for the trusted storage pool.</td>
<td>0 - 100</td>
<td>&gt;50%</td>
</tr>
<tr>
<td><code>diagnostics.brick-log-level</code></td>
<td>Changes the log-level of the bricks.</td>
<td>info</td>
<td>info/debug/warning/error/critical/none/trace</td>
</tr>
<tr>
<td><code>diagnostics.client-log-level</code></td>
<td>Changes the log-level of the clients.</td>
<td>info</td>
<td>info/debug/warning/error/critical/none/trace</td>
</tr>
<tr>
<td><code>features.read-only</code></td>
<td>Enables you to mount the entire volume as read-only for all the clients accessing it.</td>
<td>on</td>
<td>off</td>
</tr>
<tr>
<td><code>geo-replication.indexing</code></td>
<td>Use this option to automatically sync the changes in the file system from Master to Slave.</td>
<td>on</td>
<td>off</td>
</tr>
<tr>
<td><code>network.ping-timeout</code></td>
<td>The time duration for which the client waits to check if the server is responsive. When a ping timeout occurs, network disconnects between the client and server. All resources held by server on behalf of the client gets cleaned up. When network connects, all resources needs to be re-acquired before the client can resume its operations on the server. Additionally, the locks are acquired and the lock tables</td>
<td>42 seconds</td>
<td>42 seconds</td>
</tr>
</tbody>
</table>
This reconnect is a very expensive operation and should be avoided.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>nfs.export-dir</td>
<td>By default, all 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 <code>nfs.export-volumes</code> option to restrict exports only to the subdirectories specified through this option.</td>
<td>An absolute path or a comma separated list of absolute paths of subdirectories of the volumes.</td>
</tr>
<tr>
<td>nfs.export-dirs</td>
<td>By default, all subvolumes of NFS are exported as individual exports. Enabling this option allows any directory on a volumes to be exported separately.</td>
<td>on</td>
</tr>
<tr>
<td>nfs.export-volumes</td>
<td>Enables or disables exporting entire volumes. If used in conjunction with <code>nfs.export-dir</code>, can allow setting up only subdirectories as exports.</td>
<td>on</td>
</tr>
<tr>
<td>nfs.rpc-auth-allow&lt;IP-Addresses&gt;</td>
<td>Allows a comma separated list of addresses to connect to the server. By default, all clients are allowed.</td>
<td>IP address</td>
</tr>
<tr>
<td>nfs.rpc-auth-reject &lt;IP-Addresses&gt;</td>
<td>Rejects a comma separated list of addresses from connecting to the server. By default, all connections are allowed.</td>
<td>IP address</td>
</tr>
<tr>
<td>nfs.ports-insecure</td>
<td>Allows client connections from</td>
<td>on</td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Default</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>nfs.addr-namlookup</td>
<td>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 nfs.rpc-auth-* filters.</td>
<td>on</td>
<td>on</td>
</tr>
<tr>
<td>nfs.port &lt;port-number&gt;</td>
<td>Use this option on systems that need glusterFS NFS to be associated with a non-default port number.</td>
<td>1025-65535</td>
<td>38465- 38467</td>
</tr>
<tr>
<td>nfs.disable</td>
<td>Disables NFS export of individual volumes.</td>
<td>on</td>
<td>off</td>
</tr>
<tr>
<td>performance.io-thread-count</td>
<td>The number of threads in IO threads translator.</td>
<td>0 - 65</td>
<td>16</td>
</tr>
<tr>
<td>performance.cache-max-file-size</td>
<td>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 unit 64.</td>
<td>size in bytes</td>
<td>2 ^ 64-1 bytes</td>
</tr>
<tr>
<td>performance.cache-min-file-size</td>
<td>Sets the minimum file size cached by the io-cache translator. Values same as max above.</td>
<td>size in bytes</td>
<td>0 B</td>
</tr>
<tr>
<td>performance.cache-refresh-timeout</td>
<td>The cached data for a file will be retained till cache-refresh-timer expires.</td>
<td>0 - 61 seconds</td>
<td>1 second</td>
</tr>
</tbody>
</table>
timeout seconds, after which data re-validation is performed.

<table>
<thead>
<tr>
<th>Performance.Cache-Size</th>
<th>Size of the read cache.</th>
<th>Size in bytes</th>
<th>32 MB</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Server.Allow-Insecure</th>
<th>Allows 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.</th>
</tr>
</thead>
<tbody>
<tr>
<td>On</td>
<td>Off</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Server.Root-Squash</th>
<th>Prevents root users from having root privileges and assigns them the privileges of nfsnobody. This effectively squashes the power of the root user to the user nfsnobody, preventing unauthorized modification of files on the Red Hat Storage Servers.</th>
</tr>
</thead>
<tbody>
<tr>
<td>On</td>
<td>Off</td>
</tr>
</tbody>
</table>

**Important**

Red Hat recommends you 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.

You can view the changed volume settings using the `gluster volume info VOLNAME` command.

### 10.2. Expanding Volumes

You can expand volumes, as needed, while the trusted storage pool 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 Red Hat Storage volume.

Similarly, you might want to add a group of bricks to a distributed replicated volume, increasing the capacity of the Red Hat Storage volume.
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. From any server in the trusted storage pool, probe the server to which you want to add the new brick using the following command:
   
   ```bash
   # gluster peer probe HOSTNAME
   
   For example:
   
   # gluster peer probe server4
   
   Probe successful
   
   # 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:
      
      ```bash
      # 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 files are distributed to the new brick.
         
         You can use the rebalance command as described in Section 10.5, “Rebalancing Volumes”.

10.3. Shrinking Volumes

You can shrink volumes while the trusted storage pool 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.
Data residing on the brick that you are removing will no longer be accessible at the glusterFS mount point if you run the command with `force` or without any option. With `start` option, the data gets migrated to the other bricks and only the configuration information is removed - you can continue to access the data directly from the brick.

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). In a non-replicated volume, all bricks should be up to perform remove brick operation (to migrate data). In a replicated volume, at least one of the brick in the replica should be up.

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
   Remove Brick start successful
   ```

2. (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      failures
   status   -----------   -----------   -----------   -----------
   localhost                  16      16777216            52             0
         in progress
   in progress
   192.168.1.1                 13      16723211            47             0
    in progress
   ```

3. When the data migration is complete and when the `gluster volume remove-brick VOLNAME BRICK status` command displays the status as Completed, run the following command:

   ```
   # gluster volume remove-brick VOLNAME BRICK commit
   ```

   For example,

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

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

   ```
   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
```

### 10.3.1. Stopping Remove Brick Operation

**Important**

Stopping remove-brick operation is a technology preview feature. Technology Preview features are not fully supported under Red Hat subscription level agreements (SLAs), may not be functionally complete, and are not intended for production use. However, these features provide early access to upcoming product innovations, enabling customers to test functionality and provide feedback during the development process. As Red Hat considers making future iterations of Technology Preview features generally available, we will provide commercially reasonable efforts to resolve any reported issues that customers experience when using these features.

You can cancel remove-brick operation. After executing a remove-brick operation, you can choose to stop the remove-brick operation by executing `stop` command. The files which are already migrated during remove-brick operation, will not be migrated back to the same brick.

**To stop remove brick operation**

» Stop the remove brick operation using the following command:

```
# gluster volume remove-brick VOLNAME BRICK stop
```

For example:

```
# gluster volume rebalance test-volume stop
```

```
status
-------
-------
-----

617c923e-6450-4065-8e33-865e28d9428f 59 590 244
stopped
Stopped rebalance process on volume test-volume
```

### 10.4. Migrating Volumes

You can redistribute the data across bricks while the trusted storage pool is online and available.

**Note**

Before you perform `replace-brick` operation, review the known issues related to `replace-brick` operation in the Red Hat Storage 2.0 Update 4 Release Notes.
To migrate a volume

1. Make sure the new brick, server5 in this example, is successfully added to the trusted storage pool.
   For more information, see Section 7.1, “Adding Servers to Trusted Storage Pool”.

2. Migrate the data from one brick to another using the following command:
   
   ```
   # gluster volume replace-brick VOLNAME BRICK NEW-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:
   
   ```bash
   # gluster volume info VOLNAME
   ```
   
   For example, to check the volume information of new brick server5:/exp5 in test-volume:
   
   ```bash
   # 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.

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

### 10.5. Rebalancing Volumes

After expanding or shrinking (without migrating data) a volume (using the add-brick and remove-brick commands respectively), you need to rebalance the data among the servers. In a non-replicated volume, all bricks should be up to perform replace brick operation (start option). In a replicated volume, at least one of the brick in the replica should be up.

**Important**

In Red Hat Storage 2.0 Update 4 release, the fix-layout step after adding a brick has been deprecated. When a brick is added, the distributed hash table is modified automatically.

To rebalance a volume

» Start the rebalance operation on any one of the servers using the following command:

   ```bash
   # gluster volume rebalance VOLNAME start
   ```

   For example:

   ```bash
   # 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 servers using the following command:

   ```bash
   # gluster volume rebalance VOLNAME start force
   ```

   For example:
10.5.1. 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:

```
Node    Rebalanced-files          size       scanned      failures
---------         -----------   -----------   -----------   -----------   ---
localhost                 112         14567           150            0    in progress
10.16.156.72              140          2134           201            2    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:

```
Node    Rebalanced-files          size       scanned      failures
---------         -----------   -----------   -----------   -----------   ---
localhost                 112         15674           170            0    completed
10.16.156.72              140          3423           321            2    completed
```

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

```
Node    Rebalanced-files          size       scanned      failures
---------         -----------   -----------   -----------   -----------   ---
localhost                 112         15674           170            0
10.16.156.72              140          3423           321            2
```

10.5.2. Stopping Rebalance Operation

You can stop the rebalance operation, as needed.

To stop rebalance

```
# gluster volume rebalance test-volume stop
```
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 | failures |
---   | -----------------|-------|---------|----------|
localhost | 102 | 12134 | 130 | 0 |
10.16.156.72 | 110 | 2123  | 121 | 2 |

Stopped rebalance process on volume test-volume

### 10.6. 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

### 10.7. 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

### 10.8. 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 require healing.

You can view the list of files that need healing, the list of files which are currently/previosuly 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 require healing:

```bash
# gluster volume heal VOLNAME
```

For example, to trigger self-heal on files which require 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:

```bash
# gluster volume heal VOLNAME full
```

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

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

» View the list of files that need healing:

```bash
# gluster volume heal VOLNAME info
```

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

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

Brick server2:/gfs/test-volume_1
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:

```bash
# 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
Heal information on volume test-volume has been successful.
Brick server1:/gfs/test-volume_0
   Number of entries: 0

Brick server2:/gfs/test-volume_1
   Number of entries: 51
   at                path on brick
   -------------------------------
   2012-06-13 04:02:05 /dir/file.50
   2012-06-13 04:02:05 /dir/file.49
   2012-06-13 04:02:05 /dir/file.48
   2012-06-13 04:02:05 /dir/file.47
   2012-06-13 04:02:05 /dir/file.46
   2012-06-13 04:02:05 /dir/file.45
   2012-06-13 04:02:05 /dir/file.44
   ...

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

# gluster volume heal VOLNAME info healfailed
For example, to view the list of files of test-volume that are not self-healed:

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

Brick server2:/gfs/test-volume_3
   Number of entries: 72
   at                path on brick
   -------------------------------
   2012-06-13 04:02:05 /dir/file.90
   2012-06-13 04:02:05 /dir/file.95
   2012-06-13 04:02:05 /dir/file.71
   2012-06-13 04:02:05 /dir/file.67
   2012-06-13 04:02:05 /dir/file.86
   2012-06-13 04:02:05 /dir/file.55
   2012-06-13 04:02:05 /dir/file.44
   ...

» 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:
10.9. Configuring Server-Side Quorum

The configuration of quorum in a trusted storage pool determines the number of server failures that the trusted storage pool can sustain. If an additional failure occurs, the trusted storage pool becomes unavailable. It is essential that the trusted storage pool stops running, if too many server failures occur or if there is a problem with communication between the trusted storage pool nodes to prevent data loss.

Though the quorum ratio is configured at the trusted storage pool level, you can choose to or either not to enforce quorum on a particular volume by setting `cluster.server-quorum-type` volume option. For more information on this volume setting option, see Section 10.1, “Tuning Volume Options”.

Configuration of quorum is necessary to guard against network partitions in the trusted storage pool. A small set of nodes might be able to communicate together across a functioning part of a network but not be able to communicate with a different set of nodes in another part of the network. This can cause serious situations, such as split-brain in the distributed system. In a split-brain situation, at least one of the sets of nodes must stop running as a trusted storage pool.

**Configuring Server-Side Quorum**

You can configure the quorum percentage for the trusted storage pool. If the percentage of quorum is not met due to network outages, the bricks of the volume participating in quorum in those nodes are rendered offline. By default, the quorum is met if the percentage of active nodes are more than 50% to the total storage nodes. But if the user configures the ratio then quorum is considered to be met only if the percentage of active storage nodes to the total storage nodes is greater than or equal to set value.

To configure quorum:

```
# gluster volume set all cluster.server-quorum-ratio percentage%
```

For example, to set quorum to 51% on the trusted storage pool:

```
# gluster volume set all cluster.server-quorum-ratio 51%
```

Here, 51% means, at any given point of time more than half of the nodes in the trusted storage pool must be online and have network connectivity between them. If the network disconnect happens to the storage pool, then bricks running on those nodes are stopped to prevent further writes. For a two-node trusted storage pool it is important to set this value to be greater than 50% so that two nodes separated from each other do not both believe they have quorum simultaneously.
Chapter 11. 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 Red Hat Storage 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 Red Hat Storage 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`).

Both of the above types can be accessed remotely using an SSH tunnel. To use SSH, add an SSH prefix to either a file URL or glusterFS 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 form - `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.

11.1. Replicated Volumes vs Geo-replication

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

<table>
<thead>
<tr>
<th>Replicated Volumes</th>
<th>Geo-replication</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mirrors data across bricks within one trusted storage pool</td>
<td>Mirrors data across geographically distributed trusted storage pools</td>
</tr>
<tr>
<td>Provides high-availability</td>
<td>Ensures backing up of data for disaster recovery</td>
</tr>
<tr>
<td>Synchronous replication (each and every file operation is sent across all the bricks)</td>
<td>Asynchronous replication (checks for the changes in files periodically and syncs them on detecting differences)</td>
</tr>
</tbody>
</table>

11.2. 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.

- [Section 11.2.1, “Exploring Geo-replication Deployment Scenarios”](#)
- [Section 11.2.2, “Geo-replication Deployment Overview”](#)
- [Section 11.2.3, “Pre-requisite”](#)
- [Section 11.2.4, “Setting Up the Environment for Geo-replication”](#)
- [Section 11.2.5, “Setting Up the Environment for a Secure Geo-replication Slave”](#)
11.2.1. 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.
11.2.2. Geo-replication Deployment Overview

Deploying Geo-replication involves the following steps:

1. Verify that your environment matches the minimum system requirement. For more information, see Section 11.2.3, “Pre-requisite”.
2. Determine the appropriate deployment scenario. For more information, see Section 11.2.1, “Exploring Geo-replication Deployment Scenarios”.
3. Start Geo-replication on master and slave systems, as required. For more information, see Section 11.3, “Starting Geo-replication”.

11.2.3. Pre-requisite

Before deploying Geo-replication, you must ensure that both Master and Slave are Red Hat Storage instances.

11.2.4. Setting Up the Environment for Geo-replication

Time Synchronization

- On bricks of a geo-replication master volume, all the servers’ time must be uniform. You are recommended to set up NTP (Network Time Protocol) service to keep the bricks sync in time and avoid out-of-time sync effect.

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.


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). You can setup Geo-replication for SSH using any of the following methods:

Method 1 - Setup with Dedicated Service User Alias on Slave

1. Create a new user account on slave. This can be an alias user of an existing user (another user with same user ID). You can also use an alias of privileged user. For example: user@slaveshost.
2. On the node where geo-replication sessions are to be set up, run the following commands:
   
   # ssh-keygen -f /var/lib/glusterd/geo-replication/secret.pem
   Press Enter twice to avoid passphrase.
3. Run the following command on master for all the slave hosts to copy ssh-id:
   
   # ssh-copy-id -i /var/lib/glusterd/geo-replication/secret.pem.pub user@slaveshost
4. Change the shell of the user created in Step 1 to /usr/libexec/glusterfs/gsyncd by running the following command:
   
   # usermod -s /usr/libexec/glusterfs/gsyncd USERNAME

Method 2 - Using restricted SSH Key

1. Choose an user account on slave. This can be a new account or an existing one; using root account is possible. For example: user@slaveshost
2. On the node where geo-replication sessions are to be set up, run the following commands:
   
   # ssh-keygen -f /var/lib/glusterd/geo-replication/secret.pem.tmp
   Press Enter twice to avoid passphrase.
   # cat /var/lib/glusterd/geo-replication/secret.pem.tmp > /var/lib/glusterd/geo-replication/secret.pem
   # echo -n 'command="/usr/libexec/glusterfs/gsyncd"' > /var/lib/glusterd/geo-replication/secret.pem.pub
   # cat /var/lib/glusterd/geo-replication/secret.pem.tmp.pub >> /var/lib/glusterd/geo-replication/secret.pem.pub
   # rm /var/lib/glusterd/geo-replication/secret.pem.tmp
   # rm /var/lib/glusterd/geo-replication/secret.pem.tmp.pub
3. Run the following command on master for all the slave hosts:
   
   # ssh-copy-id -i /var/lib/glusterd/geo-replication/secret.pem.pub user@slaveshost

With these methods, the Slave audits commands coming from the master and the commands related to the given geo-replication session is allowed. The Slave also provides access only to the files within the slave resource which can be read or manipulated by the Master. You must not store unrestricted keys of Slave on Master to use this measure effectively.

Certain conditions are to be met and special setup is needed for the case when the chosen account is not privileged. See Section 11.2.5, “Setting Up the Environment for a Secure Geo-replication Slave” for...
details.

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

You can configure a secure slave using SSH so that master is granted a restricted access. With Red Hat Storage, you need not specify configuration parameters regarding the slave on the master-side configuration. 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 that master uses as slave resources, and the master's public key. Secure access to the slave can be established using the following options:

- Using unprivileged Red Hat Storage Over SSH
- Using IP based access control

Backward Compatibility

Your existing Geo-replication environment will work with Red Hat Storage, 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 slave.
- The following are the some exceptions where this might not work:
  - Geo-replication URLs which specify the slave resource when configuring master will include the following special characters: space, *, ?, [:.
  - Slave must have a running instance of glusterd, even if there is no Red Hat Storage volume among the mounted slave resources (that is, file tree slaves are used exclusively).

11.2.5.1. Unprivileged Red Hat Storage Slave over SSH

Geo-replication supports access to Red Hat Storage slaves through SSH using an unprivileged account (user account with non-zero uid). This method is recommended as it is more secure and it reduces the master's capabilities over slave to the minimum. This feature relies on mountbroker, an internal service of glusterd which manages the mounts for unprivileged slave accounts. You must perform additional steps to configure glusterd with the appropriate mountbroker's access control directives. The following example demonstrates this process:

To setup an auxiliary glusterFS mount for the unprivileged account:

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 owned by root and with permissions 0711. Ensure that the location where this directory is created is writable only by root but geoaccount is able to access it. For example, create a mountbroker-root directory at /var/mountbroker-root.
4. Add the following options to the glusterd volfile, assuming the name of the slave Red Hat Storage volume as slavevol:

   ```
   option mountbroker-root /var/mountbroker-root
   option mountbroker-geo-replication.geoaccount slavevol option geo-replication-log-group geogroup
   ```

If you are unable to locate the glusterd volfile at /var/lib/glusterfs/glusterd.vol, you can create a volfile containing both the default configuration and the above options and place it at /var/lib/glusterfs/.
A sample glusterd volfile along with default options:

```plaintext
volume management
type mgmt/glusterd
option working-directory /var/lib/glusterd
option transport-type socket,rdma
option transport.socket.keepalive-time 10
option transport.socket.keepalive-interval 2
option transport.socket.read-fail-log off

option 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 on Slave, you can repeat step 2. for each of them and add the following options to the volfile:

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

5. Setup Master to access Slave as geoaccount@Slave.

You can add multiple slave volumes within the same account (geoaccount) by providing comma-separated list (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:

```plaintext
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 after making changes in the configuration to effect the updates.

11.2.5.2. Using IP based Access Control

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

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

1. Set a general restriction for accessibility of file tree resources:
   ```bash
   # 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:
   ```bash
   # 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 Red Hat Storage slaves, use the following command:

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

### 11.3. Starting Geo-replication

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

- Section 11.3.1, “Starting Geo-replication”
- Section 11.3.2, “Verifying Successful Deployment”
- Section 11.3.3, “Displaying Geo-replication Status Information”
- Section 11.3.4, “Configuring Geo-replication”
- Section 11.3.5, “Stopping Geo-replication”

#### 11.3.1. Starting Geo-replication

To start Geo-replication

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

```bash
# gluster volume geo-replication MASTER SLAVE start
```

For example:

```bash
# 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 must configure the service before starting Geo-replication. For more information, see Section 11.3.4, “Configuring Geo-replication”.

**Warning**

You must not start geo-replication session on a volume that has a replace-brick operation in progress. If you start the session, there is a possibility of data loss on the slave.

#### 11.3.2. Verifying Successful Deployment

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

**To verify the status of 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
```

<table>
<thead>
<tr>
<th>MASTER</th>
<th>SLAVE</th>
<th>STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Volume1</td>
<td><a href="mailto:root@example.com">root@example.com</a>:/data/remote_dir</td>
<td>Starting...</td>
</tr>
</tbody>
</table>

### 11.3.3. 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
```

<table>
<thead>
<tr>
<th>MASTER</th>
<th>SLAVE</th>
<th>STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Volume1</td>
<td><a href="mailto:root@example.com">root@example.com</a>:/data/remote_dir</td>
<td>Starting...</td>
</tr>
</tbody>
</table>

- 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
```

<table>
<thead>
<tr>
<th>MASTER</th>
<th>SLAVE</th>
<th>STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Volume1</td>
<td>ssh://example.com:gluster://127.0.0.1:remove_volume</td>
<td>OK</td>
</tr>
<tr>
<td>Volume1</td>
<td>ssh://example.com:/data/remote_dir</td>
<td>OK</td>
</tr>
</tbody>
</table>

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 [Section 11.7, “Troubleshooting Geo-replication”](#).
- **Corrupt**: The monitor thread which is monitoring the geo-replication session has died. This situation
11.3.4. Configuring Geo-replication

To configure Geo-replication

» Use the following command at the glusterFS command line:

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

For more information about the options, see Chapter 20, 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
```

To delete a setting for a geo-replication config option, prefix the option with ! (exclamation) mark. For example, to reset the log-level to default value:

```
# gluster volume geo-replication Volume1 example.com:/data/remote_dir config '!log-level'
```

11.3.4.1. Checkpointing

Due to the asynchronous nature of geo-replication, if changes are on-going on master side, synchronization of data to slave is always in progress. Therefore, completion of synchronization of data is not possible but you can view the status of replication with respect to a particular time.

Red Hat Storage 2.0 introduces Geo-replication Checkpointing, a introspection feature. Using Checkpointing, you can get information on the progress of replication. By setting a checkpoint, the actual time is recorded as a reference timepoint, and from then on, enhanced synchronization information is available on whether the data on master as of the reference timepoint has been replicated on slave.

To configure and display geo-replication checkpoint information

» Set checkpoint to a geo-replication session

```
# gluster volume geo-replication MASTER SLAVE config checkpoint [LABEL | now]
```

For example, to set checkpoint between Volume1 and example.com:/data/remote_dir

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

geo-replication config updated successfully

» Display the status of checkpoint for a geo-replication session

```
# gluster volume geo-replication MASTER SLAVE status
```

For example, to display the status of set checkpoint between Volume1 and example.com:/data/remote_dir

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

<table>
<thead>
<tr>
<th>MASTER</th>
<th>SLAVE</th>
<th>STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Volume1 ssh://example.com:/data/remote_dir</td>
<td>OK</td>
<td>checkpoint as of 2012-06-22 11:47:01 not reached yet</td>
</tr>
</tbody>
</table>

If the set checkpoint is complete, the following status is displayed:
If you set checkpoints on a regular base, you can specify custom labels for them. For example, to set checkpoint between Volume1 and example.com:/data/remote_dir with label **NEW_ACCOUNTS_CREATED** as view its status:

```bash
# gluster volume geo-replication Volume1 example.com:/data/remote_dir config checkpoint NEW_ACCOUNTS_CREATED
geo-replication config updated successfully.
```

Delete a set checkpoint for a geo-replication session:

```bash
# gluster volume geo-replication MASTER SLAVE config '!checkpoint'
```

View the history of checkpoints for a geo-replication session:

```bash
# gluster volume geo-replication MASTER SLAVE config log-file | xargs grep checkpoint
```

11.3.5. **Stopping Geo-replication**

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

**To stop Geo-replication**

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

  ```bash
  # 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 20, Command Reference for more information about the glusterFS commands.

## 11.4. 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:

```bash
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:

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

### 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:

```bash
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:

```bash
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:
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:

   ```shell
   # gluster volume info VOLNAME
   ```

   For example:

   ```text
   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:

   ```shell
   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:
     ```shell
     Client # ls | wc -l
     100
     ```
   - On the Slave run the following command:
     ```shell
     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:

   ```shell
   # gluster volume geo-replication MASTER SLAVE start
   ```

   For example:

   ```shell
   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
   ```

### 11.5. Triggering Geo-replication Failover and Failback

**Warning**

You must engage Red Hat Profession Services before triggering geo-replication failover and failback in production environment. Contact your Red Hat representative for more information.

Red Hat Storage 2.0 supports Geo-Replication failover and failback. If the master goes down, you can trigger a failover procedure so that the slave can be replaced as the master. During this time, all I/O operations are handled by the slave until the master comes back online.
operations including writes and reads are done on the slave (now acting as master). When the original master (OM) is back online, you can trigger a failback procedure on the original slave (OS) so that it syncs the delta back to the master. The data is synced based on the time difference of xtimes (master-xtime not equal to slave-xtime).

If conflict arises while syncing data back to the master, the original master's data is replaced with the slave (now acting as master) data set. Any write operation that takes place on the master during failback is ignored.

In the commands, OM is Original Master and OS is Original Slave.

Perform the following to trigger failover and failback:

1. Start rsyncd between OS and OM to sync the missing namespaces and data by running the following:
   ```bash
   # rsync -PvSza --numeric-ids --ignore-existing /mnt/OS-VOLNAME
   PRI_HOST:/mnt/OM-VOLNAME
   ```
2. Enable blind-sync mode in OS by running the following command:
   ```bash
   # gluster volume geo-replication OS OM config special-sync-mode blind
   ```
3. Start gsyncd between OS and OM by running the following command:
   ```bash
   # gluster volume geo-replication OS OM start
   ```
4. Set a checkpoint by running the following command:
   ```bash
   # gluster volume geo-replication OS OM config checkpoint now
   ```
   Checkpoint provides status on syncing state.
5. Monitor the checkpoint till reaches status displays "OK "completed at <time of completion>".
   ```bash
   # gluster volume geo-replication OS OM status
   ```
6. Enable wrapup-sync mode by running the following command:
   ```bash
   # gluster volume geo-replication OS OM config special-sync-mode wrapup
   ```
   **Important**
   You must shutdown the user application before proceeding to Step 8 so that no writes happen on OS.
7. Repeat steps 4, 5, and 6 to know when the sync is complete. Set checkpoint again and wait for its completion.

### 11.6. 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 11.2.4, “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:
   ```bash
   # 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.
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 command is initiated. Hence it would be easier to locate the log-files when required.

11.7. Troubleshooting Geo-replication

This section describes the most common troubleshooting scenarios related to Geo-replication.

11.7.1. Locating Log Files

For every Geo-replication session, the following three log files are associated to it (four, if the slave is a Red Hat Storage 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.
11.7.2. 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
  ```

11.7.3. Synchronization is not complete

**Description**: 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 Geo-replication. After restarting, 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 10.1, “Tuning Volume Options”.

11.7.4. 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:

```
```

**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.
11.7.5. Geo-replication status displays Faulty very often

**Description:** Geo-replication displays status as faulty very often with a backtrace similar to the following:

```
```

**Solution:** This error indicates that the RPC communication between the master gsyncd module and slave gsyncd 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 and the remote machine.
- If FUSE is installed in the machine, because geo-replication module mounts the Red Hat Storage 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.

11.7.6. 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" % \ runtime Error: 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.

11.7.7. Remote gsyncd Not Found

**Description:** Master goes faulty state with the following log:

```
```

**Solution:** The steps to set up SSH connection for your geo-replication has been updated. Set up using the steps as described in Section 11.2.4, “Setting Up the Environment for Geo-replication”

11.7.8. Remote gsyncd Not Found

**Description:** You use file slave over ssh (for example: ssh://root@remote-host:/path/to/dir) and get the following connection refused error:

```
```

**Solution:** You must run glusterd on the slave even if it is a file slave.
Chapter 12. Managing Directory Quota

**Important**

Directory Quota is a technology preview feature. Technology Preview features are not fully supported under Red Hat subscription level agreements (SLAs), may not be functionally complete, and are not intended for production use. However, these features provide early access to upcoming product innovations, enabling customers to test functionality and provide feedback during the development process. As Red Hat considers making future iterations of Technology Preview features generally available, we will provide commercially reasonable efforts to resolve any reported issues that customers experience when using these features.

Directory quotas allows you to set limits on usage of disk space by directories or volumes. The storage administrators can control the disk space utilization at the directory and/or volume levels by setting limits to allocatable disk space at any level in the volume and directory hierarchy. This is particularly useful in cloud deployments to facilitate utility billing model.

**Note**

Red Hat Storage supports only setting hard limits. Once the hard limit is set, it cannot exceed the limit and attempts to use more disk space beyond the set limit is 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 12.3, "Setting or Replacing Disk Limit."

### 12.1. 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
  ```
12.2. 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 translator on test-volume:

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

12.3. 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 on a directory using the following command:

```
# gluster volume quota VOLNAME limit-usage /directory limit-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
```

Set the disk limit on a volume using the following command:

```
# gluster volume quota VOLNAME limit-usage / limit-value
```

For example, to set limit on test-volume:

```
# gluster volume quota test-volume limit-usage / 100GB
Usage limit has been set on /
```

In this example, / refers to Red Hat Storage mount point. If the mount point is /mnt/rhs1 then the quota limit set on / is considered for /mnt/rsh1.

Note

In a multi-level directory hierarchy, the strictest disk limit will be considered for enforcement.

12.4. 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:

```bash
# gluster volume quota test-volume list
```

<table>
<thead>
<tr>
<th>Path</th>
<th>Limit</th>
<th>Set Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>/Test/data</td>
<td>10 GB</td>
<td>6 GB</td>
</tr>
<tr>
<td>/Test/data1</td>
<td>10 GB</td>
<td>4 GB</td>
</tr>
</tbody>
</table>

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

```bash
# gluster volume quota `VOLNAME` list /directory name
```

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

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

<table>
<thead>
<tr>
<th>Path</th>
<th>Limit</th>
<th>Set Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>/Test/data</td>
<td>10 GB</td>
<td>6 GB</td>
</tr>
</tbody>
</table>

## 12.5. Updating the Timeout of Size Cache

For performance reasons, quota caches the directory sizes on client. You can set the timeout duration 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 disable directory size caching on client side.

**To update the timeout of size cache**

- Update the timeout of size cache using the following command:
  ```bash
  # gluster volume set `VOLNAME` features.quota-timeout `value`
  ```

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

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

## 12.6. 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:
  ```bash
  # 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
Chapter 13. Monitoring your Red Hat Storage Workload

Monitoring volumes helps in capacity planning and performance tuning tasks of the Red Hat Storage volume. You can monitor the Red Hat Storage volumes on different parameters and use those system outputs to identify and troubleshoot issues.

You can use Volume Top and Profile commands to view the performance and identify bottlenecks 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.

Note

If you restart the server process, the existing Profile and Top information will be reset.

13.1. Running Volume Profile Command

Red Hat Storage Volume Profile command provides an interface to get the per-brick or NFS server I/O information for each File Operation (FOP) of a volume. This information helps in identifying bottlenecks in the storage system.

This section describes how to run Red Hat Storage Volume Profile command by performing the following operations:

- Section 13.1.1, "Start Profiling"
- Section 13.1.2, "Displaying the I/O Information"
- Section 13.1.3, "Stop Profiling"

13.1.1. 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

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

To display I/O information:

» Display the I/O information of the bricks of a volume 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:

<table>
<thead>
<tr>
<th>Block Size:</th>
<th>1b+</th>
<th>32b+</th>
<th>64b+</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read:</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Write:</td>
<td>908</td>
<td>28</td>
<td>8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Block Size:</th>
<th>128b+</th>
<th>256b+</th>
<th>512b+</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read:</td>
<td>0</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>Write:</td>
<td>5</td>
<td>23</td>
<td>16</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Block Size:</th>
<th>1024b+</th>
<th>2048b+</th>
<th>4096b+</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read:</td>
<td>0</td>
<td>52</td>
<td>17</td>
</tr>
<tr>
<td>Write:</td>
<td>15</td>
<td>120</td>
<td>846</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Block Size:</th>
<th>8192b+</th>
<th>16384b+</th>
<th>32768b+</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read:</td>
<td>52</td>
<td>8</td>
<td>34</td>
</tr>
<tr>
<td>Write:</td>
<td>234</td>
<td>134</td>
<td>286</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Block Size:</th>
<th>65536b+</th>
<th>131072b+</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read:</td>
<td>118</td>
<td>622</td>
</tr>
<tr>
<td>Write:</td>
<td>1341</td>
<td>594</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>%-latency</th>
<th>Avg- latency</th>
<th>Min- Latency</th>
<th>Max- Latency</th>
<th>calls</th>
<th>Fop</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.82</td>
<td>1132.28</td>
<td>21.00</td>
<td>800970.00</td>
<td>4575</td>
<td>WRITE</td>
</tr>
<tr>
<td>5.70</td>
<td>156.47</td>
<td>9.00</td>
<td>665085.00</td>
<td>39163</td>
<td>readdirp</td>
</tr>
<tr>
<td>11.35</td>
<td>315.02</td>
<td>9.00</td>
<td>1433947.00</td>
<td>38698</td>
<td>lookup</td>
</tr>
<tr>
<td>11.88</td>
<td>1729.34</td>
<td>21.00</td>
<td>2569638.00</td>
<td>7382</td>
<td>fxattrop</td>
</tr>
<tr>
<td>47.35</td>
<td>104235.02</td>
<td>2485.00</td>
<td>7789367.00</td>
<td>488</td>
<td>fsync</td>
</tr>
</tbody>
</table>

Duration : 335

BytesRead : 94505058

BytesWritten : 195571980
```

» Display the I/O information of a NFS server of the specified volume using the following command:
# gluster volume profile VOLNAME info nfs
For example, to see the I/O information of NFS server:
```
# gluster volume profile test-volume info nfs
NFS Server : localhost

Cumulative Stats:

<table>
<thead>
<tr>
<th>Block Size</th>
<th>32768b+</th>
<th>65536b+</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of Reads</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>No. of Writes</td>
<td>1000</td>
<td>1000</td>
</tr>
<tr>
<td>%-latency</td>
<td>--------</td>
<td>---------</td>
</tr>
<tr>
<td>Avg-latency</td>
<td>--------</td>
<td>---------</td>
</tr>
<tr>
<td>Min-Latency</td>
<td>--------</td>
<td>---------</td>
</tr>
<tr>
<td>Max-Latency</td>
<td>--------</td>
<td>---------</td>
</tr>
<tr>
<td>No. of calls</td>
<td>--------</td>
<td>---------</td>
</tr>
<tr>
<td>Fop</td>
<td>--------</td>
<td>---------</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------</td>
<td>---------</td>
</tr>
<tr>
<td>STATFS</td>
<td>0.01</td>
<td>194.00 us</td>
</tr>
<tr>
<td>FSTAT</td>
<td>0.60</td>
<td>346.00 us</td>
</tr>
<tr>
<td>SETATTR</td>
<td>1.63</td>
<td>67.00 us</td>
</tr>
<tr>
<td>ACCESS</td>
<td>1.94</td>
<td>58.00 us</td>
</tr>
<tr>
<td>TRUNCATE</td>
<td>2.55</td>
<td>52.00 us</td>
</tr>
<tr>
<td>TRUNCATE</td>
<td>2.85</td>
<td>88.00 us</td>
</tr>
<tr>
<td>FLUSH</td>
<td>4.47</td>
<td>60.00 us</td>
</tr>
<tr>
<td>READDIRP</td>
<td>5.02</td>
<td>1723.00 us</td>
</tr>
<tr>
<td>READDIRP</td>
<td>25.42</td>
<td>101.00 us</td>
</tr>
<tr>
<td>READDIRP</td>
<td>55.52</td>
<td>124.00 us</td>
</tr>
</tbody>
</table>

Duration: 7074 seconds
Data Read: 0 bytes
Data Written: 102400000 bytes

Interval 1 Stats:

<table>
<thead>
<tr>
<th>Block Size</th>
<th>32768b+</th>
<th>65536b+</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of Reads</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>No. of Writes</td>
<td>1000</td>
<td>1000</td>
</tr>
<tr>
<td>%-latency</td>
<td>--------</td>
<td>---------</td>
</tr>
<tr>
<td>Avg-latency</td>
<td>--------</td>
<td>---------</td>
</tr>
<tr>
<td>Min-Latency</td>
<td>--------</td>
<td>---------</td>
</tr>
<tr>
<td>Max-Latency</td>
<td>--------</td>
<td>---------</td>
</tr>
<tr>
<td>No. of calls</td>
<td>--------</td>
<td>---------</td>
</tr>
<tr>
<td>Fop</td>
<td>--------</td>
<td>---------</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------</td>
<td>---------</td>
</tr>
<tr>
<td>STATFS</td>
<td>0.01</td>
<td>194.00 us</td>
</tr>
<tr>
<td>FSTAT</td>
<td>0.60</td>
<td>346.00 us</td>
</tr>
<tr>
<td>SETATTR</td>
<td>1.63</td>
<td>67.00 us</td>
</tr>
<tr>
<td>ACCESS</td>
<td>1.94</td>
<td>58.00 us</td>
</tr>
<tr>
<td>TRUNCATE</td>
<td>2.55</td>
<td>52.00 us</td>
</tr>
<tr>
<td>TRUNCATE</td>
<td>2.85</td>
<td>88.00 us</td>
</tr>
<tr>
<td>FLUSH</td>
<td>4.47</td>
<td>60.00 us</td>
</tr>
<tr>
<td>READDIRP</td>
<td>5.02</td>
<td>1723.00 us</td>
</tr>
<tr>
<td>READDIRP</td>
<td>25.41</td>
<td>101.00 us</td>
</tr>
<tr>
<td>READDIRP</td>
<td>55.53</td>
<td>124.00 us</td>
</tr>
</tbody>
</table>

Duration: 330 seconds
```
### 13.1.3. 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:
  
  ```bash
  # gluster volume profile VOLNAME stop
  ```

  For example, to stop profiling on `test-volume`:

  ```bash
  # gluster volume profile test-volume stop
  ```

  Profiling stopped on test-volume

### 13.2. Running Volume Top Command

Red Hat Storage 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 **Top** commands:

- Section 13.2.1, “Viewing Open File Descriptor Count and Maximum File Descriptor Count”
- Section 13.2.2, “Viewing Highest File Read Calls”
- Section 13.2.3, “Viewing Highest File Write Calls”
- Section 13.2.4, “Viewing Highest Open Calls on Directory”
- Section 13.2.5, “Viewing Highest Read Calls on Directory”
- Section 13.2.6, “Viewing List of Read Performance”
- Section 13.2.7, “Viewing List of Write Performance”

#### 13.2.1. Viewing Open File Descriptor Count and Maximum File Descriptor Count

You can view current open file descriptor count and the list of files that are currently being accessed on the brick. It also displays the maximum open file descriptor count of files that are 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 the open file descriptor metrics of all the bricks belonging to the volume is displayed.

**To view open file descriptor count and maximum file descriptor count:**

- View open file descriptor count and maximum file descriptor count using the following command:

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

  For example, to view open file descriptor count and maximum file descriptor count on brick `server:/export` of `test-volume` and list top 10 open calls:
13.2.2. 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:

```bash
# gluster volume top VOLNAME read [nfs | 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========

<table>
<thead>
<tr>
<th>read call count</th>
<th>filename</th>
</tr>
</thead>
<tbody>
<tr>
<td>116</td>
<td>/clients/client0/~dmtmp/SEED/LARGE.FIL</td>
</tr>
<tr>
<td>64</td>
<td>/clients/client0/~dmtmp/SEED/MEDIUM.FIL</td>
</tr>
<tr>
<td>54</td>
<td>/clients/client2/~dmtmp/SEED/LARGE.FIL</td>
</tr>
<tr>
<td>54</td>
<td>/clients/client6/~dmtmp/SEED/LARGE.FIL</td>
</tr>
<tr>
<td>54</td>
<td>/clients/client5/~dmtmp/SEED/LARGE.FIL</td>
</tr>
<tr>
<td>54</td>
<td>/clients/client0/~dmtmp/SEED/LARGE.FIL</td>
</tr>
<tr>
<td>54</td>
<td>/clients/client3/~dmtmp/SEED/LARGE.FIL</td>
</tr>
<tr>
<td>54</td>
<td>/clients/client4/~dmtmp/SEED/LARGE.FIL</td>
</tr>
<tr>
<td>54</td>
<td>/clients/client9/~dmtmp/SEED/LARGE.FIL</td>
</tr>
<tr>
<td>54</td>
<td>/clients/client8/~dmtmp/SEED/LARGE.FIL</td>
</tr>
</tbody>
</table>

13.2.3. 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 [nfs | 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
```
13.2.4. Viewing Highest Open Calls on Directory

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:
  ```
13.2.5. 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:

```bash
# gluster volume top VOLNAME readdir [nfs | brick BRICK-NAME] [list-cnt cnt]
```

For example, to view highest directory read calls on brick `server:/export/` of `test-volume`:

---

```bash
# gluster volume top test-volume opendir brick server:/export/ list-cnt 10
Brick: server:/export/dir1

===========Directory open stats========
<table>
<thead>
<tr>
<th>Opendir count</th>
<th>directory name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1001</td>
<td>/clients/client0/~dmtmp</td>
</tr>
<tr>
<td>454</td>
<td>/clients/client8/~dmtmp</td>
</tr>
<tr>
<td>454</td>
<td>/clients/client2/~dmtmp</td>
</tr>
<tr>
<td>454</td>
<td>/clients/client6/~dmtmp</td>
</tr>
<tr>
<td>454</td>
<td>/clients/client5/~dmtmp</td>
</tr>
<tr>
<td>454</td>
<td>/clients/client9/~dmtmp</td>
</tr>
<tr>
<td>443</td>
<td>/clients/client0/~dmtmp/ PARADOX</td>
</tr>
<tr>
<td>408</td>
<td>/clients/client1/~dmtmp</td>
</tr>
<tr>
<td>408</td>
<td>/clients/client7/~dmtmp</td>
</tr>
<tr>
<td>402</td>
<td>/clients/client4/~dmtmp</td>
</tr>
</tbody>
</table>
```
13.2.6. Viewing List of Read Performance

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.

This command will initiate a read for the specified count and block size and measures the corresponding throughput directly on the back-end export (bypassing glusterFS processes).

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] [nfs | 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:
13.2.7. Viewing List of Write Performance

You can view list of write throughput of files on each brick or nfs server. 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 write for the specified count and block size and measures the corresponding throughput directly on back-end export (bypassing glusterFS processes). To view list of write performance on each brick:

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

```bash
# gluster volume top VOLNAME write-perf [bs blk-size count count] [nfs | 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:

```bash
# 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

```bash
# 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========

<table>
<thead>
<tr>
<th>write throughput (MBps)</th>
<th>filename</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1170.00</td>
<td>/clients/client0/~dmtmp/SEED/SMALL.FIL</td>
<td>-2012-05-09 15:39:09.171494</td>
</tr>
<tr>
<td>1008.00</td>
<td>/clients/client6/~dmtmp/SEED/LARGE.FIL</td>
<td>-2012-05-09 15:39:09.73189</td>
</tr>
<tr>
<td>949.00</td>
<td>/clients/client0/~dmtmp/SEED/MEDIUM.FIL</td>
<td>-2012-05-09 15:38:36.927426</td>
</tr>
<tr>
<td>936.00</td>
<td>/clients/client0/~dmtmp/SEED/LARGE.FIL</td>
<td>-2012-05-09 15:38:36.933177</td>
</tr>
<tr>
<td>897.00</td>
<td>/clients/client5/~dmtmp/SEED/MEDIUM.FIL</td>
<td>-2012-05-09 15:39:09.33628</td>
</tr>
<tr>
<td>897.00</td>
<td>/clients/client6/~dmtmp/SEED/MEDIUM.FIL</td>
<td>-2012-05-09 15:39:09.27713</td>
</tr>
<tr>
<td>885.00</td>
<td>/clients/client0/~dmtmp/SEED/SMALL.FIL</td>
<td>-2012-05-09 15:38:36.924271</td>
</tr>
<tr>
<td>528.00</td>
<td>/clients/client5/~dmtmp/SEED/LARGE.FIL</td>
<td>-2012-05-09 15:39:09.81893</td>
</tr>
<tr>
<td>516.00</td>
<td>/clients/client6/~dmtmp/ACCESS/FASTENER.MDB</td>
<td>-2012-05-09 15:39:01.797317</td>
</tr>
</tbody>
</table>
```

### 13.3. Listing Volumes

You can list all volumes in the trusted storage pool.

**To list all volumes**

» List all volume using the following command:

```bash
# gluster volume list
```

For example, to list all volumes in the trusted storage pool:

```bash
# gluster volume list

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>test-volume</td>
<td>volume1</td>
<td>volume2</td>
</tr>
<tr>
<td></td>
<td>volume3</td>
<td></td>
</tr>
</tbody>
</table>
```

### 13.4. Displaying Volume Information

112
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
  ```

### 13.5. 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. You can use the
following options 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 file descriptor tables of the volume.
- **inode** - Dumps the inode tables of the volume.
- **history** - Dumps the event history of the volume

**To perform a volume statedump**

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

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

  For example, to display statedump of test-volume:

  ```bash
  # 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`.

- You can change the directory of the statedump file using the following command:

  ```bash
  # gluster volume set VOLNAME server.statedump-path path
  ```

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

  ```bash
  # 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:

  ```bash
  # gluster volume info VOLNAME
  ```

**13.6. Displaying Volume Status**

You can display the status information about a specific volume, brick or all volumes, as needed. Status information can be used to understand the current status of the brick, NFS processes, self-heal daemon and overall file system. Status information can also be used to monitor and debug the volume information. You can view status of the volume along with the 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 file descriptor 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:

```bash
# gluster volume status [all|VOLNAME [nfs | shd | BRICKNAME]] [detail [clients | mem | inode | fd |callpool]]
```

For example, to display information about test-volume:

```bash
# gluster volume status test-volume
Status of volume: test-volume
Gluster process Port Online Pid
------------------------------------------------------------
Brick arch:/export/rep1 24010 Y 18474
Brick arch:/export/rep2 24011 Y 18479
NFS Server on localhost 38467 Y 18486
Self-heal Daemon on localhost N/A Y 18491
```

The self-heal daemon status will be displayed only for replicated volumes.

Display information about all volumes using the following command:

```bash
# gluster volume status all
```

```bash
# gluster volume status all
Status of volume: test
Gluster process Port Online Pid
-----------------------------------------------------------
Brick 192.168.56.1:/export/test 24009 Y 29197
NFS Server on localhost 38467 Y 18486
```

```bash
# gluster volume status detail
```

```bash
# gluster volume status detail
Status of volume: test-vol
Brick : Brick arch:/exp
Port : 24012
Online : Y
Pid : 18649
File System : ext4
Device : /dev/sda1
Mount Options : rw,relatime,user_xattr,acl,commit=600,barrier=1,data=ordered
Inode Size : 256
Disk Space Free : 22.1GB
Total Disk Space : 46.5GB
Inode Count : 3055616
Free Inodes : 2577164
```

Detail information is not available for NFS and self-heal daemon.
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
```

Clients information is not available for self-heal daemon.

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:
Display the inode tables of the volume using the following command:

```bash
# gluster volume status VOLNAME inode
```

For example, to display the inode tables of the test-volume:
Display the open file descriptor tables of the volume using the following command:

```
# gluster volume status VOLNAME fd
```

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

```
# gluster volume status test-volume fd
```

<table>
<thead>
<tr>
<th>FD Entry</th>
<th>PID</th>
<th>RefCount</th>
<th>Flags</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>26311</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>26310</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>26310</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>26311</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

FD information is not available for NFS and self-heal daemon.

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 callpool

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
Chapter 14. Managing Red Hat Storage Volume Life-Cycle Extensions

Red Hat Storage allows automation of operations by user written scripts. For every operation, you can execute a pre and a post script.

Pre Scripts - These scripts are run before the occurrence of the event. You can write a script to automate activities like manage system-wide services. For example, you can write a script to stop exporting the SMB share corresponding to the volume before you stop the volume.

Post Scripts - These scripts are run after execution of the event. For example, you can write a script to export the SMB share corresponding to the volume after you start the volume.

You can run scripts for the following events:

- Creating a volume
- Starting a volume
- Adding a brick
- Removing a brick
- Tuning volume options
- Stopping a volume
- Deleting a volume

Naming Convention

While creating file names of your scripts, you must follow the naming convention followed in your underlying file system like XFS.

Note

To enable the script, the name of the script must start with a S. Scripts run in lexicographic order of their names.

14.1. Location of Scripts

This section provides information on the folders where the scripts must be placed. When you create a trusted storage pool, the following directories are created:

- /var/lib/glusterd/hooks/1/create/
- /var/lib/glusterd/hooks/1/delete/
- /var/lib/glusterd/hooks/1/start/
- /var/lib/glusterd/hooks/1/stop/
- /var/lib/glusterd/hooks/1/set/
- /var/lib/glusterd/hooks/1/add-brick/
- /var/lib/glusterd/hooks/1/remove-brick/

After creating a script, you must ensure to save the script in its respective folder on all the nodes of the trusted storage pool. The location of the script dictates whether the script needs to be executed before or after an event. Scripts are provided with a command line arguments --volname=VOLNAME. Command specific additional arguments are provided for the following volume operations:
Start volume
- --first=yes, if the volume is the first to be started
- --first=no, for otherwise

Stop volume
- --last=yes, if the volume is to be stopped last.
- --last=no, for otherwise

Set volume
- -o key=value

For every key, value is specified in volume set command.

14.2. Prepackaged Scripts

Red Hat provides scripts to export Samba (SMB) share when you start a volume and to remove the share when you stop the volume. These scripts are available at: /var/lib/glusterd/hooks/1/start/post and /var/lib/glusterd/hooks/1/stop/pre

By default, the scripts are enabled.

When you start a volume using the following command:

```
# gluster volume start VOLNAME
```

The `S30samba-start.sh` script performs the following:

1. Adds Samba share configuration details of the volume to the `smb.conf` file
2. Mounts the volume through FUSE and adds an entry in `/etc/fstab` for the same.
3. Restarts Samba to run with updated configuration

When you stop the volume using the following command:

```
# gluster volume stop VOLNAME
```

The `S30samba-stop.sh` script performs the following:

1. Removes the Samba share details of the volume from the `smb.conf` file
2. Unmounts the FUSE mount point and removes the corresponding entry in `/etc/fstab`
3. Restarts Samba to run with updated configuration
Part III. Red Hat Storage Administration on Public Cloud
Chapter 15. Launching Red Hat Storage Server for Public Cloud

Red Hat Storage Server for Public Cloud is a pre-integrated, pre-verified and ready to run Amazon Machine Image (AMI) that provides a fully POSIX compatible highly available scale-out NAS and object storage solution for the Amazon Web Services (AWS) public cloud infrastructure.

**Note**

For information on obtaining access to AMI, see [https://access.redhat.com/knowledge/articles/145693](https://access.redhat.com/knowledge/articles/145693).

This chapter describes how to launch Red Hat Storage instance on Amazon Web Services.

### 15.1. Launching Red Hat Storage Instance

This section describes how to launch Red Hat Storage instance on Amazon Web Services.

**To launch the Red Hat Storage Instance**


2. Login to Amazon Web Services. The Amazon Web Services main screen appears.

3. Click the **Amazon EC2** tab. The Amazon EC2 Console Dashboard appears.

4. Click the **Launch Instance** button. The **Choose an AMI** step of the Request Instances
Wizard appears.

5. Click the Select button for the corresponding AMI. The Instance Details screen appears.

6. Choose Large using the Instance Type menu, and click Continue. The Instance Details screen continues.

7. Accept the default settings, and click Continue. The Instance Details screen continues.
8. Type a name for the instance in the **Value** field for **Name** in **Key** field, and click **Continue**. You can use this name later to verify that the instance is operating correctly.

The **Create Key Pair** screen appears.

9. Choose an existing key pair or create a new key pair, and click **Continue**. The **Configure Firewall** screen appears.

10. Select a security group from the **Security Groups** field, and click **Continue**. You must ensure to open the following TCP port numbers in the selected security group:
11. Review your settings, and click **Launch**. A screen appears indicating that the instance is launching.

12. Click **Close**. The Amazon EC2 Console Dashboard appears.

### 15.2. Verifying that Red Hat Storage Instance is running

You can verify that Red Hat Storage instance is running by performing a remote login to the Red Hat Storage instance and issuing a command.

**To verify that Red Hat is running**

1. On the Amazon Web Services home page, click the **Amazon EC2** tab. The **Amazon EC2 Console Dashboard** appears.
2. Click the *Instances* link in the *Instances* section on the left. The *My Instances* screen appears showing your current instances.

3. Check the Status column and verify that the instance is running. A yellow circle indicates a status of pending while a green circle indicates that the instance is running.

Click the instance and verify the details displayed in the *Description* tab.
4. Note the domain name in the **Public DNS** field. You can use this domain to perform a remote login to the instance.

5. Using SSH and the domain from the previous step, login to the Red Hat Amazon Machine Image instance. Use the key pair that you selected or created when launching the instance.

   Example:
   
   Enter the following in command line:

   ```
   # ssh -i rhs-aws.pem root ec2-23-20-52-123.compute-1.amazonaws.com
   ```

6. At the command line, enter the following command:

   ```
   # service glusterd status
   ```
   Verify that the command indicates that the glusterd daemon is running.
Chapter 16. Provisioning Storage

Amazon Elastic Block Storage (EBS) is designed specifically for use with Amazon EC2 instances. Amazon EBS provides storage that behaves like a raw, unformatted, external block device. Red Hat supports 8 EBS volumes assembled into a software RAID 0 array for use as a single brick.

Important

Single EBS volume exhibits inconsistent I/O performance. Red Hat supported configuration is 8 Amazon EBS volumes of equal size on software RAID 0 (stripe) attached to a brick enables consistent I/O performance. You can create a brick ranging from 8 GB to 8 TB. For example, if you create a brick of 128 GB, you must create 8 Amazon EBS volumes of size 16 GB each and then create RAID 0 (stripe). Other configurations are not supported by Red Hat.

To add EBS volumes and provision storage

1. Create an EBS volume using the following command:
   
   # ec2-create-volume -s SIZE -z ZONE
   
   For example,
   
   # ec2-create-volume -s 16 -z us-east-1a vol-239289
   
   This command creates a volume of 16 GB and returns the volume ID.

2. Use the generated volume ID to attach the created volume to an existing Red Hat Storage instance, using the following command:

   # ec2-attach-volume volumeid -i instanceid -d DEVICE
   
   For example,
   
   # ec2-attach-volume vol-239289 -i i-343833 -d /dev/sdf1
   
   The device /dev/sdf1 appears as /dev/xvdf1 in the instance.

   Important

   You must repeat the above steps to add remaining 7 Amazon EBS volumes to the existing instances.

3. Amazon EBS volumes are assembled in to a RAID volume in support of configuration as a brick. Assemble software RAID 0 (stripe) using the following command:

   # mdadm --create arrayname --level=0 --raid-devices=8 list of all devices
   
   For example, to create a software raid of eight volumes

   # mdadm --create /dev/md0 --level=0 --raid-devices=8 /dev/xvdf1 /dev/xvdf2 /dev/xvdf3 /dev/xvdf4 /dev/xvdf5 /dev/xvdf6 /dev/xvdf7 /dev/xvdf8
   
   # mdadm --examine --scan > /etc/mdadm.conf

4. Tag the attached EBS volumes using the following command:

   ec2-create-tags VOLID \
   --tag Domain=gluster \

130
Table 16.1. Field Description

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VOLID</td>
<td>EBS volume ID</td>
</tr>
<tr>
<td>DEVICE</td>
<td>Device path of the EBS volume in the instance</td>
</tr>
<tr>
<td>INSTANCE_ID</td>
<td>Instance Id to which the volume is attached</td>
</tr>
<tr>
<td>UUID</td>
<td>UUID of the array device. You can obtain the UUID of the array device by running `# mdadm --detail DEVICE</td>
</tr>
</tbody>
</table>

For example, to tag `/dev/xvdf1` device, run the following command:

```
# ec2-create-tags vol-239289
--tag Domain=gluster
--tag Device=/dev/xvdf1
--tag Instance=i-343833
--tag Array=e35641fc:621a9fa8:276a0ee4:a1bc6b5f
```

Repeat Step 4 to tag all eight EBS volumes.

5. Create a LVM Logical Volume (LV) by running the following commands:

```
# pvcreate /dev/md0
# vgcreate glustervg /dev/md0
# vgchange -a y glustervg
# lvcreate -a y -l 100%VG -n glusterlv glustervg
```

Here `glustervg` is the name of the volume group and `glusterlv` is the name of the logical volume. This LVM logical volume created over EBS RAID volumes is used as Red Hat Storage bricks.


6. Format the LVM LV using the following command:

```
# mkfs.xfs -i size=512 DEVICE
```

For example, to format `/dev/glustervg/glusterlv` run the following command:

```
# mkfs.xfs -i size=512 /dev/glustervg/glusterlv
```

7. Mount the device using the following commands:

```
# mkdir -p /export/glusterlv
# mount /dev/glustervg/glusterlv /export/glusterlv
```

8. Add the device entry to `/etc/fstab` to mount the device automatically on every reboot. Run the following command to automatically add the device entry to `fstab`

```
# echo "'/dev/glustervg/glusterlv /export/glusterlv xfs defaults 0 2" >> /etc/fstab
```

After you provision the storage, you can now use the mount point as brick with existing volumes or create new volumes. For more information on creating volumes, see [Chapter 8, Setting up Red Hat Storage Volumes](https://access.redhat.com/knowledge/docs/en-US/Red_Hat_Enterprise_Linux/6/html/Logical_Volume_Manager_Administration/index.html#hbfs-01000)
Chapter 17. Stopping and Restarting Red Hat Storage Instance

When you stop and restart a Red Hat Storage instance, a new IP address and hostname is assigned by Amazon Web Services and it looses its association with the virtual hardware. Hence, the newly restarted Red Hat Storage instance will not be part of the storage pool and causes disruptions to the trusted storage pool. If the restarted Red Hat Storage instance is not part of trusted storage pool, add it to the trusted pool as described in Section 7.1, “Adding Servers to Trusted Storage Pool”.

Rebooting the Red Hat Storage instance preserves the IP address and hostname and does not loose its association with the virtual hardware. This does not cause any disruptions to the trusted storage pool.
Part IV. Data Access with Other Interfaces
Chapter 18. Managing Unified File and Object Storage

Unified File and Object Storage 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.

Red Hat Storage is based on glusterFS, an open source distributed file system. Unified File and Object Storage is built upon OpenStack’s Object Storage Swift. OpenStack 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 back-end file system for OpenStack Swift. It also leverages on OpenStack 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 SMB. 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 Red Hat Storage 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 an open source software for creating redundant, scalable object storage using clusters of standardized servers to store petabytes of accessible data. It is not a file system or real-time data storage system, but rather a long-term storage system for a more permanent type of static data that can be retrieved, leveraged, and updated.

Figure 18.1. Unified File and Object Storage Architecture
18.1. Components of Object Storage

The major components of Object Storage are:

Proxy Server

All REST requests to the Unified File and Object Storage 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 Unified File and Object Storage 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.

Unified File and Object Storage implements accounts as Red Hat Storage volumes. So, when a user is granted read/write permission on an account, it means that the user has access to all the data available on that Red Hat Storage 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.

18.2. Advantages of using Unified File and Object Storage

The following are the advantages of using Unified File and Object Storage:

Note

When you install Red Hat Storage 2.0, Unified File and Object Storage is automatically installed, by default.
No limit on upload and download files sizes as compared to OpenStack Swift which limits the object size to 5 GB.
A unified view of data across NAS and Object Storage technologies.
Using Red Hat Storage's Unified File and Object Storage has other advantages like the following:
- High availability
- Scalability
- Replication
- Elastic Volume Management

18.3. Pre-requisites
You must start **memcached** service using the following command:

```
# service memcached start
```

**Ports**

The following ports must be open for Unified File and Object Storage to work:

- 6000 - Object Server
- 6001 - Container Server
- 6002 - Account Server
- Proxy Server
  - 443 - for HTTPS request
  - 8080 - for HTTP request

18.4. Configuring Unified File and Object Storage

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

18.4.1. 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]
```

The account name should be the name of the Red Hat Storage volume. When a user is granted read/write permission on an account, it means that the user has access to all the data available on that Red Hat Storage volume.
For example,

```
user_test_tester = testing .admin
```

**Important**

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 18.5.3.6, “Setting ACLs on Container”.

### 18.4.2. 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, Unified File and Object Storage is configured to support HTTP protocol and uses temporary authentication to authenticate the HTTP requests.

### 18.4.3. Configuring Authentication System

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

### 18.4.4. 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 18.1. Proxy Server - Configurable Default Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bind_ip</td>
<td>127.0.0.1</td>
<td>IP Address for server to bind.</td>
</tr>
<tr>
<td>bind_port</td>
<td>8080</td>
<td>Port for server to bind.</td>
</tr>
<tr>
<td>swift_dir</td>
<td>/etc/swift</td>
<td>Swift configuration directory.</td>
</tr>
<tr>
<td>workers</td>
<td>1</td>
<td>Number of workers to fork.</td>
</tr>
<tr>
<td>user</td>
<td>swift</td>
<td>Swift user.</td>
</tr>
<tr>
<td>cert_file</td>
<td></td>
<td>Path to the ssl.crt file.</td>
</tr>
<tr>
<td>key_file</td>
<td></td>
<td>Path to the ssl.key file.</td>
</tr>
</tbody>
</table>

Note

You must change the bind_ip default value to external IP address of that machine for server to bind. Otherwise, from a remote client, you cannot use native swift-client (/usr/bin/swift) and you will get wrong URL while building X-Storage-Url.
### Table 18.2. Proxy Server - Configurable Server Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>use</td>
<td><code>egg:swift#container</code></td>
<td>Entry point for paste.deploy for the container server.</td>
</tr>
<tr>
<td>log_name</td>
<td><code>proxy-server</code></td>
<td>Label used when logging.</td>
</tr>
<tr>
<td>log_facility</td>
<td><code>LOG_LOCAL0</code></td>
<td>Syslog log facility.</td>
</tr>
<tr>
<td>log_level</td>
<td><code>INFO</code></td>
<td>Log level.</td>
</tr>
<tr>
<td>log_headers</td>
<td><code>True</code></td>
<td>If True, log headers in each request.</td>
</tr>
<tr>
<td>recheck_account_existence</td>
<td><code>60</code></td>
<td>Cache timeout in seconds to send memcached for account existence.</td>
</tr>
<tr>
<td>recheck_container_existence</td>
<td><code>60</code></td>
<td>Cache timeout in seconds to send memcached for container existence.</td>
</tr>
<tr>
<td>object_chunk_size</td>
<td><code>65536</code></td>
<td>Chunk size to read from object servers.</td>
</tr>
<tr>
<td>client_chunk_size</td>
<td><code>65536</code></td>
<td>Chunk size to read from clients.</td>
</tr>
<tr>
<td>memcache_servers</td>
<td><code>127.0.0.1:11211</code></td>
<td>Comma separated list of memcached servers ip:port.</td>
</tr>
<tr>
<td>node_timeout</td>
<td><code>10</code></td>
<td>Request timeout to external services.</td>
</tr>
<tr>
<td>client_timeout</td>
<td><code>60</code></td>
<td>Timeout to read one chunk from a client.</td>
</tr>
<tr>
<td>conn_timeout</td>
<td><code>0.5</code></td>
<td>Connection timeout to external services.</td>
</tr>
<tr>
<td>error_suppression_interval</td>
<td><code>60</code></td>
<td>Time in seconds that must elapse since the last error for a node to be considered no longer error limited.</td>
</tr>
<tr>
<td>error_suppression_limit</td>
<td><code>10</code></td>
<td>Error count to consider a node error limited.</td>
</tr>
<tr>
<td>allow_account_management</td>
<td><code>false</code></td>
<td>Whether account <code>PUT</code>s and <code>DELETE</code>s are even callable.</td>
</tr>
</tbody>
</table>
Enabling Distributed Caching with Memcached

When Object Storage is deployed on two or more machines, not all nodes in your trusted storage pool are used. Installing a load balancer enables you to utilize all the nodes in your trusted storage pool by distributing the proxy server requests equally to all storage nodes. You must configure the proxy servers on all the nodes to use a distributed memcached to share the authentication token across all the storage nodes. Edit the `memcache_servers` config option in the `proxy-server.conf` and list all memcached servers. Following is an example listing the memcached servers in the `proxy-server.conf` file.

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

The port number on which the memcached server is listening is 11211. You must ensure to use the same sequence for all configuration files.

### 18.4.5. 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 to 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 18.3. Object Server - Configurable Default Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>swift_dir</td>
<td>/etc/swift</td>
<td>Swift configuration directory.</td>
</tr>
<tr>
<td>devices</td>
<td>/srv/node</td>
<td>Mount parent directory where devices are mounted.</td>
</tr>
<tr>
<td>mount_check</td>
<td>true</td>
<td>Whether or not check if the devices are mounted to prevent accidentally writing to the root device.</td>
</tr>
<tr>
<td>bind_ip</td>
<td>0.0.0.0</td>
<td>IP Address for server to bind.</td>
</tr>
<tr>
<td>bind_port</td>
<td>6000</td>
<td>Port for server to bind.</td>
</tr>
<tr>
<td>workers</td>
<td>1</td>
<td>Number of workers to fork.</td>
</tr>
</tbody>
</table>

### Table 18.4. Object Server - Configurable Server Options

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

### 18.4.6. 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 18.5. Container Server - Configurable Default Options**

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

**Table 18.6. Container Server - Configurable Server Options**

<table>
<thead>
<tr>
<th>Option</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>use</td>
<td>egg:swift#container</td>
<td>Entry point for paste.deploy for the container server.</td>
</tr>
<tr>
<td>log_name</td>
<td>container-server</td>
<td>Label used when logging.</td>
</tr>
<tr>
<td>log_facility</td>
<td>LOG_LOCAL0</td>
<td>Syslog log facility.</td>
</tr>
<tr>
<td>log_level</td>
<td>INFO</td>
<td>Logging level.</td>
</tr>
<tr>
<td>node_timeout</td>
<td>3</td>
<td>Request timeout to external services.</td>
</tr>
<tr>
<td>conn_timeout</td>
<td>0.5</td>
<td>Connection timeout to external services.</td>
</tr>
</tbody>
</table>
18.4.7. 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 Unified File and Object Storage, each Red Hat Storage 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>
<thead>
<tr>
<th>Option</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>swift_dir</td>
<td>/etc/swift</td>
<td>Swift configuration directory.</td>
</tr>
<tr>
<td>devices</td>
<td>/srv/node</td>
<td>Mount parent directory where devices are mounted.</td>
</tr>
<tr>
<td>mount_check</td>
<td>true</td>
<td>Whether or not check if the devices are mounted to prevent accidentally writing to the root device.</td>
</tr>
<tr>
<td>bind_ip</td>
<td>0.0.0.0</td>
<td>IP address for server to bind.</td>
</tr>
<tr>
<td>bind_port</td>
<td>6002</td>
<td>Port for server to bind.</td>
</tr>
<tr>
<td>workers</td>
<td>1</td>
<td>Number of workers to fork.</td>
</tr>
<tr>
<td>user</td>
<td>swift</td>
<td>Swift user.</td>
</tr>
</tbody>
</table>

Table 18.7. Account Server - Configurable Default Options
Table 18.8. Account Server - Configurable Server Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>use</td>
<td>egg:swift#container</td>
<td>Entry point for paste.deploy for the container server.</td>
</tr>
<tr>
<td>log_name</td>
<td>account-server</td>
<td>Label used when logging.</td>
</tr>
<tr>
<td>log_facility</td>
<td>LOG_LOCAL0</td>
<td>Syslog log facility.</td>
</tr>
<tr>
<td>log_level</td>
<td>INFO</td>
<td>Logging level.</td>
</tr>
</tbody>
</table>

18.4.8. Starting and Stopping Server
You must start the server manually when the system reboots and whenever you update or 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
```

To automatically start the gluster-swift services every time the system boots, run the following command:

```
# chkconfig memcached on
# chkconfig gluster-swift-proxy on
# chkconfig gluster-swift-account on
# chkconfig gluster-swift-container on
# chkconfig gluster-swift-object on
```

18.5. 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.

18.5.1. 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_tkde3ad38b087b49bbac0494f7600a554
X-Auth-Token: AUTH_tkde3ad38b087b49bbac0494f7600a554
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:

```
```

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**

By default, the authentication tokens are valid for a 24 hour period. However, you can configure the validity of authentication token.

### 18.5.2. Working with Accounts

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

#### 18.5.2.1. 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:

**Table 18.9. Parameters - Container Information**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>limit</td>
<td>Limits the number of results to ( n )th value.</td>
</tr>
<tr>
<td>marker</td>
<td>Returns object names greater in value than the specified marker.</td>
</tr>
<tr>
<td>format</td>
<td>Specify either <strong>JSON</strong> or <strong>XML</strong> to return the respective serialized response.</td>
</tr>
</tbody>
</table>

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_tkd3ad38b087b49bbbac0494f7600a554' https://example.storage.com:443/v1/AUTH_test -k
```

18.5.2.2. 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:
18.5.3. Working with Containers
This section describes the list of operations you can perform at the container level of the URL.

18.5.3.1. 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:

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

For example,

```bash
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:

```bash
curl -v -X PUT -H 'X-Auth-Token: AUTH_tkd3ad38b087b49bbbac0494f7600a554'
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 a same name already exists, the 202 status code is displayed.

18.5.3.2. Displaying Objects of a Container
You can list the objects of a container using the GET command. You can use the following optional parameters with the GET request to refine the results:
Table 18.10. Parameters - Container Objects

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>limit</td>
<td>Limits the number of results to ( n )th value.</td>
</tr>
<tr>
<td>marker</td>
<td>Returns object names greater in value than the specified marker.</td>
</tr>
<tr>
<td>prefix</td>
<td>Displays the results limited to object names beginning with the substring ( x ).</td>
</tr>
<tr>
<td>path</td>
<td>Returns the object names nested in the pseudo path.</td>
</tr>
<tr>
<td>format</td>
<td>Specify either JSON or XML to return the respective serialized response.</td>
</tr>
<tr>
<td>delimiter</td>
<td>Returns all the object names nested in the container.</td>
</tr>
</tbody>
</table>

To display objects of a container

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

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

For example,

```plaintext
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:

```plaintext
curl -v -X GET -H 'X-Auth-Token: AUTH_tkd3ad38b087b49bbbac0494f7600a554' https://example.storage.com:443/v1/AUTH_test/images -k
```

18.5.3.3. 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:
For example,

```
HEAD /v1/AUTH_test/images HTTP/1.1
Host: example.storage.com
X-Auth-Token: AUTH_tkd3ad38b087b49babbac0494f7600a554

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_tkd3ad38b087b49babbac0494f7600a554'
https://example.storage.com:443/v1/AUTH_test/images -k
```

### 18.5.3.4. 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_tkd3ad38b087b49babbac0494f7600a554

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_tkd3ad38b087b49babbac0494f7600a554'
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.

18.5.3.5. 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:

```
```

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.

18.5.3.6. 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:
Examples of invalid ACLs:

```
.r: *
.r:* , bobs_account, sues_account:sue
bobs_account, sues_account:sue
```

By default, allowing read access via `.r` will not allow listing objects in the container but allows retrieving objects from the container. Use the `rlistings` directive to turn on listings. 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_tkde3ad38b087b49bbac8494f7600a554' -H 'X-Container-Read: .r:*' https://example.storage.com:443/v1/AUTH_test/images
```

### 18.5.4. 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.

#### 18.5.4.1. 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 MD5 checksum 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,
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_tkd3ad38b087b49bbbac0494f7600a554' 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.

18.5.4.1.1. 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

18.5.4.2. 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:

```plaintext
COPY /<apiversion>/<account>/<container>/<sourceobject> HTTP/1.1
Host: <storage URL>
X-Auth-Token: < authentication-token-key>
Destination: /<container>/<destinationobject>
```

For example,

```plaintext
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:

```plaintext
curl -v -X COPY -H 'X-Auth-Token: AUTH_tkd3ad38b087b49bbbac0494f7600a554' -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:

```plaintext
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:

```plaintext
curl -v -X PUT -H 'X-Auth-Token: AUTH_tkd3ad38b087b49bbbac0494f7600a554'
-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.
18.5.4.3. 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_tkd3ad38b087b49bcbac049f7600a554

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: 8a964ee2a6e88be344f36c22562a6486
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_tkd3ad38b087b49bcbac049f7600a554' 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.

18.5.4.4. 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,
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_tkd3ad38b087b49bbac0494f7600a554' 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.

18.5.4.5. 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_tkd3ad38b087b49bbac0494f7600a554
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:

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.

### 18.5.4.6. Deleting Object

You can use the **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.
Chapter 19. Managing Hadoop Compatible Storage

Important

Hadoop Compatible Storage is a technology preview feature. Technology Preview features are not fully supported under Red Hat subscription level agreements (SLAs), may not be functionally complete, and are not intended for production use. However, these features provide early access to upcoming product innovations, enabling customers to test functionality and provide feedback during the development process. As Red Hat considers making future iterations of Technology Preview features generally available, we will provide commercially reasonable efforts to resolve any reported issues that customers experience when using these features.

Red Hat Storage 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 Red Hat Storage seamlessly. This new functionality opens up data within Hadoop deployments to any file-based or object-based application.

Note

When you install Red Hat Storage 2.0, Hadoop Compatible Storage is automatically installed, by default.

19.1. Architecture Overview

The following diagram illustrates Hadoop integration with Red Hat Storage:

19.2. Advantages
The following are the advantages of Hadoop Compatible Storage with Red Hat Storage:

- Provides simultaneous file-based and object-based access within Hadoop.
- Eliminates the centralized metadata server.
- Provides compatibility with MapReduce applications and code rewrite is not required.
- Provides a fault tolerant file system.

### 19.3. Preparing to Install Hadoop Compatible Storage

This section provides information on pre-requisites and a list of dependencies that will be installed during installation of Hadoop compatible storage.

#### 19.3.1. 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 trusted storage pool. For more information on installing, configuring, and running Hadoop, see [http://hadoop.apache.org/common/docs/r0.15.2/cluster_setup.html](http://hadoop.apache.org/common/docs/r0.15.2/cluster_setup.html).
- Java Runtime Environment.
  To install Java Runtime Environment, run `# yum install java-1.6.0-openjdk` command.

### 19.4. Configuring Hadoop Compatible Storage

This section describes how to configure Hadoop Compatible Storage in your storage environment and verify that it is functioning correctly.

To configure Hadoop compatible storage:

1. Edit the `core-site.xml` file available at `/usr/share/java/conf`. The following is the sample `core-site.xml` file:
The following table lists the fields of core-site.xml file that you can configure:
### Table 19.1. Configurable Fields

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fs.default.name</td>
<td>glusterfs://192.168.1.36:9000</td>
<td>Any hostname in the trusted storage pool as the server and any port number.</td>
</tr>
<tr>
<td>fs.glusterfs.volname</td>
<td>volume-dist-rep</td>
<td>Red Hat Storage volume to mount.</td>
</tr>
<tr>
<td>fs.glusterfs.mount</td>
<td>/mnt/glusterfs</td>
<td>The directory used to fuse mount the volume.</td>
</tr>
<tr>
<td>fs.glusterfs.server</td>
<td>192.168.1.36</td>
<td>Any hostname or IP address on the trusted storage pool.</td>
</tr>
<tr>
<td>quick.slave.io</td>
<td>Off</td>
<td>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. As a result, read performance improves and jobs run faster.</td>
</tr>
</tbody>
</table>

**Note**

This option is not tested widely.

2. Copy `glusterfs-0.20.2-0.1.jar` and `core-site.xml` files to Hadoop's `lib/` and `conf/` directory respectively using the following commands:

```
# cp /usr/share/java/glusterfs-0.20.2-0.1.jar $HADOOP_HOME/lib/
```

```
# cp /usr/share/java/conf/core-site.xml $HADOOP_HOME/conf/
```

### 19.5. 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.
19.6. Troubleshooting Hadoop Compatible Storage
This section describes the most common troubleshooting issues related to Hadoop Compatible Storage.

19.6.1. Time Sync
Running MapReduce job may throw exceptions if the time is out-of-sync on the hosts in the trusted storage pool.

Solution: Sync the time on all hosts using ntpd program.
Part V. Appendices
Chapter 20. Command Reference

This section describes the available commands and includes the following section:

- **gluster Command**
  - Gluster Console Manager (command line interpreter)
- **glusterd Daemon**
  - GlusterFS elastic volume management daemon

### 20.1. 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. You can run the gluster command on any export server. The command enables administrators to perform cloud operations such as creating, expanding, shrinking, rebalancing, and migrating volumes without needing to schedule server downtime.

#### COMMANDS

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Volume</strong></td>
<td>Displays information about all volumes, or the specified volume.</td>
</tr>
<tr>
<td>gluster volume info [all</td>
<td>VOLNAME]</td>
</tr>
<tr>
<td>gluster volume create NEW-VOLNAME [stripe COUNT] [replica COUNT] [transport tcp</td>
<td>rdma</td>
</tr>
<tr>
<td>gluster volume delete VOLNAME</td>
<td>Deletes the specified volume.</td>
</tr>
<tr>
<td>gluster volume start VOLNAME</td>
<td>Starts the specified volume.</td>
</tr>
<tr>
<td>gluster volume stop VOLNAME [force]</td>
<td>Stops the specified volume.</td>
</tr>
<tr>
<td>gluster volume list</td>
<td>Lists all volumes in the trusted storage pool.</td>
</tr>
<tr>
<td>gluster volume help</td>
<td>Displays help for the volume command.</td>
</tr>
</tbody>
</table>

<p>| <strong>Brick</strong> | Adds the specified bricks to the specified volume. |
| gluster volume add-brick VOLNAME [&lt;stripe|replica&gt; &lt;COUNT&gt;] NEW-BRICK ... | |
| gluster volume remove-brick VOLNAME [(replica | COUNT] NEW-BRICK ... | Removes the specified bricks from the specified volume. |</p>
<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>gluster volume set VOLNAME &lt;KEY&gt; &lt;VALUE&gt;</code></td>
<td>Sets the specified option and value for the specified volume.</td>
</tr>
<tr>
<td><code>gluster volume sync HOSTNAME [all] VOLNAME</code></td>
<td>Syncs the volume information from a peer.</td>
</tr>
<tr>
<td><code>gluster volume reset VOLNAME [option] [force]</code></td>
<td>Resets specified or all reconfigured options to its default value.</td>
</tr>
<tr>
<td>`gluster volume replace-brick VOLNAME (BRICK NEW-BRICK) {start</td>
<td>pause</td>
</tr>
<tr>
<td><code>gluster volume replace-brick VOLNAME BRICK NEW-BRICK pause</code></td>
<td>Starts migration of data from one brick to another.</td>
</tr>
<tr>
<td><code>gluster volume replace-brick VOLNAME BRICK NEW-BRICK pause</code></td>
<td>Pauses migration of data.</td>
</tr>
<tr>
<td><code>gluster volume replace-brick VOLNAME BRICK NEW-BRICK pause</code></td>
<td>Aborts migration of data.</td>
</tr>
<tr>
<td><code>gluster volume replace-brick VOLNAME BRICK NEW-BRICK pause</code></td>
<td>Displays the status of data migration.</td>
</tr>
<tr>
<td><code>gluster volume replace-brick VOLNAME BRICK NEW-BRICK commit</code></td>
<td>Commits the migration of data from one brick to another.</td>
</tr>
</tbody>
</table>

**Profile**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>gluster volume profile VOLNAME start</code></td>
<td>Starts the profiling to view the file operation information for each brick.</td>
</tr>
<tr>
<td><code>gluster volume profile VOLNAME info [nfs]</code></td>
<td>Displays the I/O information for each brick or NFS servers.</td>
</tr>
<tr>
<td><code>gluster volume profile VOLNAME stop</code></td>
<td>Stops profiling the specified volume.</td>
</tr>
</tbody>
</table>

**Quota**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>gluster volume quota VOLNAME enable [path] [value]</code></td>
<td>Enables quota to set disk limits.</td>
</tr>
<tr>
<td><code>gluster volume quota VOLNAME disable [path] [value]</code></td>
<td>Disables quota set on the volume.</td>
</tr>
<tr>
<td><code>gluster volume quota VOLNAME limit-usage [path] [value]</code></td>
<td>Sets the disk limit.</td>
</tr>
<tr>
<td><code>gluster volume quota VOLNAME list [path] [value]</code></td>
<td>Display disk limit information set on the directories.</td>
</tr>
<tr>
<td><code>gluster volume quota VOLNAME remove [path] [value]</code></td>
<td>Deletes the quota limit set on the directory.</td>
</tr>
</tbody>
</table>

**Status**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>`gluster volume status [all</td>
<td>VOLNAME [nfs</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>gluster volume status all</code></td>
<td>Displays information about all volumes.</td>
</tr>
<tr>
<td><code>gluster volume status VOLNAME detail</code></td>
<td>Displays additional information about the bricks.</td>
</tr>
<tr>
<td><code>gluster volume status VOLNAME clients</code></td>
<td>Displays the list of clients accessing the volumes.</td>
</tr>
<tr>
<td><code>gluster volume status VOLNAME mem</code></td>
<td>Displays the memory usage and memory pool details of the bricks.</td>
</tr>
<tr>
<td><code>gluster volume status VOLNAME inode</code></td>
<td>Displays the inode tables of the volume.</td>
</tr>
<tr>
<td><code>gluster volume status VOLNAME fd</code></td>
<td>Displays the open file descriptor tables of the volume.</td>
</tr>
<tr>
<td><code>gluster volume status VOLNAME callpool</code></td>
<td>Displays the pending calls of the volume.</td>
</tr>
</tbody>
</table>

**Self-Heal**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>gluster volume heal VOLNAME [options]</code></td>
<td>Self-heal commands on specified volume.</td>
</tr>
<tr>
<td><code>gluster volume heal VOLNAME</code></td>
<td>Triggers self-heal only on the files which requires healing.</td>
</tr>
<tr>
<td><code>gluster volume heal VOLNAME full</code></td>
<td>Triggers self-heal on all the files of a volume.</td>
</tr>
<tr>
<td><code>gluster volume heal VOLNAME info</code></td>
<td>Displays the list of files that needs healing.</td>
</tr>
<tr>
<td><code>gluster volume heal VOLNAME info healed</code></td>
<td>Displays the list of files that are self-healed.</td>
</tr>
<tr>
<td><code>gluster volume heal VOLNAME info heal-failed</code></td>
<td>Displays the list of files of a particular volume on which the self-heal failed.</td>
</tr>
</tbody>
</table>

**Statedump**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>`gluster volume statedump VOLNAME [nfs] [all</td>
<td>mem</td>
</tr>
<tr>
<td><code>gluster volume set VOLNAME server.statedump-path path</code></td>
<td>Change the directory of the statedump file.</td>
</tr>
</tbody>
</table>

**Locks**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>`gluster volume clear-locks VOLNAME path kind {blocked</td>
<td>granted</td>
</tr>
</tbody>
</table>

**Top**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>`gluster volume top VOLNAME open [nfs</td>
<td>brick BRICK-NAME] [list-cnt cnt]`</td>
</tr>
<tr>
<td>`gluster volume top VOLNAME read [nfs</td>
<td>brick BRICK-NAME] [list-cnt cnt]`</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>gluster volume top VOLNAME write [nfs</td>
<td>brick BRICK-NAME] [list-cnt cnt]</td>
</tr>
<tr>
<td>gluster volume top VOLNAME opendir [nfs</td>
<td>brick BRICK-NAME] [list-cnt cnt]</td>
</tr>
<tr>
<td>gluster volume top VOLNAME readdir [nfs</td>
<td>brick BRICK-NAME] [list-cnt cnt]</td>
</tr>
<tr>
<td>gluster volume top VOLNAME read-perf [bs blk-size count count] [brick BRICK-NAME] [list-cnt cnt]</td>
<td>Displays the list of read performance on each brick of the specified brick/NFS server of the volume.</td>
</tr>
<tr>
<td>gluster volume top VOLNAME write-perf [bs blk-size count count] [nfs</td>
<td>brick BRICK-NAME] [list-cnt cnt]</td>
</tr>
<tr>
<td><strong>Rebalance</strong></td>
<td>Rebalance Operations</td>
</tr>
<tr>
<td>gluster volume rebalance VOLNAME [fix-layout] {start</td>
<td>stop</td>
</tr>
<tr>
<td>gluster volume rebalance VOLNAME start</td>
<td>Starts rebalancing the specified volume.</td>
</tr>
<tr>
<td>gluster volume rebalance VOLNAME stop</td>
<td>Stops rebalancing the specified volume.</td>
</tr>
<tr>
<td>gluster volume rebalance VOLNAME status</td>
<td>Displays the rebalance status of the specified volume.</td>
</tr>
<tr>
<td><strong>Log</strong></td>
<td>Rotates the log file for corresponding volume/brick.</td>
</tr>
<tr>
<td>volume log rotate VOLNAME [BRICK]</td>
<td>Rotates the log file for corresponding volume/brick.</td>
</tr>
<tr>
<td><strong>Peer</strong></td>
<td></td>
</tr>
<tr>
<td>peer probe HOSTNAME</td>
<td>Probes the specified peer.</td>
</tr>
<tr>
<td>peer detach HOSTNAME</td>
<td>Detaches the specified peer.</td>
</tr>
<tr>
<td>peer status</td>
<td>Displays the status of peers.</td>
</tr>
<tr>
<td>peer help</td>
<td>Displays help for the peer command.</td>
</tr>
<tr>
<td><strong>Geo-replication</strong></td>
<td></td>
</tr>
<tr>
<td>volume geo-replication MASTER SLAVE start</td>
<td>Start geo-replication between the hosts specified by MASTER and SLAVE. You can specify a local master volume as :VOLNAME.</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>volume geo-replication MASTER SLAVE stop</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>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</td>
</tr>
</tbody>
</table>
### volume geo-replication MASTER SLAVE log-rotate

Rotates the log file of a particular master-slave session.

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

Configure geo-replication options between the hosts specified by MASTER and SLAVE.

- **gluster-log-file LOGFILE**
  The path to the geo-replication glusterfs log file.

- **gluster-log-level LOGFILELEVEL**
  The log level for glusterfs processes.

- **log-file LOGFILE**
  The path to the geo-replication log file.

- **log-level LOGFILELEVEL**
  The log level for geo-replication.

- **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.

- **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.

- **checkpoint [LABEL | now]**
  Sets the checkpoint with the given option LABEL. If the option is set as now, then the current time will be used as label.

### Other

- **help**
  Display the command options.

- **quit**
  Exit the gluster command line interface.

### FILES
/var/lib/glusterd/*

### SEE ALSO
20.2. glusterd Daemon

NAME

glusterd - GlusterFS 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

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-l=LOGFILE, --log-file=LOGFILE</td>
<td>Files to use for logging (the default is /usr/local/var/log/glusterfs/glusterfs.log).</td>
</tr>
<tr>
<td>-L=LOGLEVEL, --log-level=LOGLEVEL</td>
<td>Logging severity. Valid options are TRACE, DEBUG, INFO, WARNING, ERROR and CRITICAL (the default is INFO).</td>
</tr>
<tr>
<td>--debug</td>
<td>Runs the program in debug mode. This option sets --no-daemon, --log-level to DEBUG, and --log-file to console.</td>
</tr>
<tr>
<td>-N, --no-daemon</td>
<td>Runs the program in the foreground.</td>
</tr>
</tbody>
</table>

FILES

/var/lib/glusterd/*

SEE ALSO

mount.glusterfs(8), glusterfs(8), gluster(8), glustefs(8)
Chapter 21. Troubleshooting

This section describes how to manage logs and most common troubleshooting scenarios related to Red Hat Storage.

21.1. Managing Red Hat Storage Logs

This section describes how to manage Red Hat Storage logs by performing the following operation:

- Rotating Logs

21.1.1. 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.

21.2. Troubleshooting File Locks

In Red Hat Storage 2.0 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 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 13.5, “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
   ```

   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`.

2. Clear the entry lock using the following command:

```bash
# gluster volume clear-locks VOLNAME path kind granted entry basename
```

The following are the sample contents of the statedump file indicating entry lock (entrylk). Ensure that those are stale locks and no resources own them.

```
[xlator.features.locks.vol-locks.inode]
path=/
mandatory=0
entrylk-count=1
lock-dump.domain.domain=vol-replicate-0
xlator.feature.locks.lock-dump.domain.entrylk.entrylk[0](ACTIVE)=type=ENTRYLK_WRLCK on basename=file1,
pid = 714782904, owner=ffffff2a3c7f0000, transport=0x20e0670, , granted at Mon Feb 27 16:01:01 2012
```

For example, to clear the entry lock on `file1` of test-volume:

```bash
# gluster volume clear-locks test-volume / path kind granted entry file1
Volume clear-locks successful
test-volume-locks: entry blocked locks=0 granted locks=1
```

3. Clear the inode lock using the following command:

```bash
# gluster volume clear-locks VOLNAME path kind granted inode range
```

The following are the sample contents of the statedump file indicating there is an inode lock (inodelk). Ensure that those are stale locks and no resources own them.

```
[conn.2.bound_xl./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
inodelk.inodelk[0](ACTIVE)=type=WRITE, whence=0, start=0, len=0, pid = 714787072, owner=00ffff2a3c7f0000, transport=0x20e0670, , granted at Mon Feb 27 16:01:01 2012
```

For example, to clear the inode lock on `file1` of test-volume:

```bash
# gluster  volume clear-locks test-volume /file1 kind granted inode 0,0-0
Volume clear-locks successful
test-volume-locks: inode blocked locks=0 granted locks=1
```
4. Clear the granted POSIX lock using the following command:

```
# gluster volume clear-locks VOLNAME path kind granted posix range
```

The following are the sample contents of the statedump file indicating there is a granted POSIX lock. Ensure that those are stale locks and no resources own them.

```
xlator.features.locks.vol1-locks.inode
path=/file1
mandatory=0
posixlk-count=15
posixlk.posixlk[0](ACTIVE)=type=WRITE, whence=0, start=8, len=1, pid = 23848, 
owner=d824f04c60c3c73c, transport=0x120b370, , blocked at Mon Feb 27 16:01:01 2012,
, granted at Mon Feb 27 16:01:01 2012
posixlk.posixlk[1](ACTIVE)=type=WRITE, whence=0, start=7, len=1, pid = 1, 
owner=30404152462d436c-69656e7431, transport=0x11eb4f0, , granted at Mon Feb 27 16:01:01 2012
posixlk.posixlk[2](BLOCKED)=type=WRITE, whence=0, start=8, len=1, pid = 1, 
owner=30404152462d436c-69656e7431, transport=0x11eb4f0, , granted at Mon Feb 27 16:01:01 2012
posixlk.posixlk[3](ACTIVE)=type=WRITE, whence=0, start=6, len=1, pid = 12776, 
owner=a36bb0aea0258969, transport=0x120a4e0, , granted at Mon Feb 27 16:01:01 2012
```

For example, to clear the granted POSIX lock on `file1` of test-volume:

```
# gluster volume clear-locks test-volume /file1 kind granted posix 0,8-1
Volume clear-locks successful
```

5. Clear the blocked POSIX lock using the following command:

```
# gluster volume clear-locks VOLNAME path kind blocked posix range
```

The following are the sample contents of the statedump file indicating there is a blocked POSIX lock. Ensure that those are stale locks and no resources own them.
For example, to clear the blocked POSIX lock on file1 of test-volume:

```
# gluster volume clear-locks test-volume /file1 kind blocked posix 0,0-1
Volume clear-locks successful
test-volume-locks: posix blocked locks=28 granted locks=0
test-volume-locks: posix blocked locks=1 granted locks=0
No locks cleared.
```

6. Clear all POSIX locks using the following command:

```
# gluster volume clear-locks VOLNAME path kind all posix range
```

The following are the sample contents of the statedump file indicating there is all POSIX locks. Ensure that those are stale locks and no resources own them.
For example, to clear the all POSIX locks on file1 of test-volume:

```
# gluster volume clear-locks test-volume /file1 kind all posix 0,0-1
Volume clear-locks successful

test-volume-locks: posix blocked locks=1 granted locks=0
No locks cleared.

test-volume-locks: posix blocked locks=4 granted locks=1
```

You can perform statedump on test-volume again to verify that all the above locks are cleared.
## Revision History

<table>
<thead>
<tr>
<th>Revision 1-65.400</th>
<th>2013-10-31</th>
<th>Rüdiger Landmann</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rebuild with publican 4.0.0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Revision 1-65</th>
<th>Thu May 30 2013</th>
<th>Divya Muntimadugu</th>
</tr>
</thead>
<tbody>
<tr>
<td>Added a note in Rebalancing Volumes section.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Revision 1-64</th>
<th>Wed Apr 17 2013</th>
<th>Divya Muntimadugu</th>
</tr>
</thead>
<tbody>
<tr>
<td>Updated Configuring Server-Side Quorum section.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Revision 1-61</th>
<th>Tue Apr 2 2013</th>
<th>Divya Muntimadugu</th>
</tr>
</thead>
<tbody>
<tr>
<td>Updated Rebalancing Volumes section.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Revision 1-59</th>
<th>Thu Mar 28 2013</th>
<th>Divya Muntimadugu</th>
</tr>
</thead>
<tbody>
<tr>
<td>Updated the guide with instructions on installing Native Client.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Revision 1-58</th>
<th>Tue Mar 26 2013</th>
<th>Divya Muntimadugu</th>
</tr>
</thead>
<tbody>
<tr>
<td>Added a note on replace-brick operation.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Revision 1-56</th>
<th>Thu Mar 21 2013</th>
<th>Divya Muntimadugu</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bug fixes</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Revision 1-55</th>
<th>Tue Mar 19 2013</th>
<th>Divya Muntimadugu</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bug fixes</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Revision 1-52</th>
<th>Wed Mar 13 2013</th>
<th>Divya Muntimadugu</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bug fixes</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Revision 1-46</th>
<th>Fri Mar 01 2013</th>
<th>Divya Muntimadugu</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bug fixes</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Revision 1-45</th>
<th>Sat Feb 23 2013</th>
<th>Divya Muntimadugu</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bug fixes</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Revision 1-44</th>
<th>Wed Feb 13 2013</th>
<th>Divya Muntimadugu</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bug fixes</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Revision 1-42</th>
<th>Fri Jan 11 2013</th>
<th>Divya Muntimadugu</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bug fixes</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Revision 1-41</th>
<th>Thu Jan 03 2013</th>
<th>Divya Muntimadugu</th>
</tr>
</thead>
<tbody>
<tr>
<td>Updated guide with Server-Side Quorum Configuration steps.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Revision 1-40</th>
<th>Wed Dec 12 2012</th>
<th>Divya Muntimadugu</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bug fixes</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Revision 1-39</th>
<th>Wed Dec 05 2012</th>
<th>Divya Muntimadugu</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bug fixes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Revision 1-38</td>
<td>Mon Dec 03 2012</td>
<td>Divya Muntimadugu</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>Bug fixes and added content for <strong>eager-lock</strong> - volume set option.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Revision 1-36</th>
<th>Thu Nov 22 2012</th>
<th>Divya Muntimadugu</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bug fixes</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Revision 1-35</th>
<th>Mon Nov 12 2012</th>
<th>Divya Muntimadugu</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bug fixes</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Revision 1-34</th>
<th>Tue Nov 06 2012</th>
<th>Divya Muntimadugu</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bug fixes</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Revision 1-33</th>
<th>Wed Oct 17 2012</th>
<th>Divya Muntimadugu</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bug fixes</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Revision 1-31</th>
<th>Mon Oct 08 2012</th>
<th>Divya Muntimadugu</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bug fixes</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Revision 1-30</th>
<th>Wed Oct 03 2012</th>
<th>Divya Muntimadugu</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bug fixes</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Revision 1-29</th>
<th>Mon Sep 24 2012</th>
<th>Divya Muntimadugu</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bug fixes</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Revision 1-28</th>
<th>Fri Sep 14 2012</th>
<th>Divya Muntimadugu</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bug fixes</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Revision 1-27</th>
<th>Wed Sep 12 2012</th>
<th>Divya Muntimadugu</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bug fixes</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Revision 1-26</th>
<th>Tue Sep 04 2012</th>
<th>Divya Muntimadugu</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bug fixes</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Revision 1-25</th>
<th>Fri Aug 24 2012</th>
<th>Divya Muntimadugu</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bug fixes</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Revision 1-24</th>
<th>Mon Aug 06 2012</th>
<th>Divya Muntimadugu</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bug fixes</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Revision 1-3</th>
<th>Mon Jul 23 2012</th>
<th>Divya Muntimadugu</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bug fixes</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Revision 1-2</th>
<th>Tue Jul 17 2012</th>
<th>Divya Muntimadugu</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bug fixes</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Revision 1-1</th>
<th>Tue Jun 26 2012</th>
<th>Divya Muntimadugu</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version for 2.0 GA release</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>