

**Red Hat Storage Server
Administration
Student Workbook
Red Hat Storage 2.0
Release en-1-20130201**





RED HAT STORAGE SERVER ADMINISTRATION

Red Hat Storage 2.0 RH236

Red Hat Storage Server Administration

Edition 1

Author	Wander Boessenkool
Editor	George Hacker
Editor	Steven Bonneville

Copyright © 2013 Red Hat, Inc.

The contents of this course and all its modules and related materials, including handouts to audience members, are Copyright © 2013 Red Hat, Inc.

No part of this publication may be stored in a retrieval system, transmitted or reproduced in any way, including, but not limited to, photocopy, photograph, magnetic, electronic or other record, without the prior written permission of Red Hat, Inc.

This instructional program, including all material provided herein, is supplied without any guarantees from Red Hat, Inc. Red Hat, Inc. assumes no liability for damages or legal action arising from the use or misuse of contents or details contained herein.

If you believe Red Hat training materials are being used, copied, or otherwise improperly distributed please e-mail training@redhat.com or phone toll-free (USA) +1 (866) 626-2994 or +1 (919) 754-3700.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, JBoss, Hibernate, 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 registered trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

All other trademarks are the property of their respective owners.

Contributors: Victor Costea, Vedaraman Shankar, Niels de Vos

Document Conventions	v
Notes and Warnings	v
Introduction	vii
Welcome to class!	vii
About Red Hat Enterprise Linux	vii
Additional Red Hat Enterprise Linux Software	viii
Contacting Red Hat Technical Support	ix
About This Course	xi
Red Hat Storage Server Administration	xi
Structure of the Course	xi
Orientation to the Classroom Network	xi
Internationalization	xiii
Language Support	xiii
System-wide Default Language	xiii
Per-user Language Selection	xiii
Input Methods	xiv
Language Codes Reference	xiv
1. Introduction to Red Hat Storage Server	1
Features	2
Use Cases	4
Concepts & Terminology	6
Hardware Requirements	8
2. Explore the Classroom Environment	11
Exploring the Classroom	12
Unit Test	15
3. Installing Red Hat Storage Server	19
Installing Red Hat Storage Server on Premise	20
Installing Red Hat Storage Server on Public Cloud	26
4. Basic Configuration	29
Building a Trusted Storage Pool	30
5. Volume Types	37
Different Volume Types	38
Unit Test	43
6. Clients	47
Native Client	48
NFS Clients	51
CIFS Clients	53
Volume Options	56
Unit Test	58
7. ACLs and Quotas	61
POSIX ACLs	62
Quotas	64
Unit Test	66
8. Extending Volumes	69
Growing Volumes	70
Unit Test	73

9. IP Failover	77
IP Failover with CTDB	78
Unit Test	81
10. Geo-Replication	85
Configuring Geo-Replication	86
11. Unified File and Object Storage	93
Introduction to Swift	94
Configuring Swift	96
Testing Swift	99
12. Troubleshooting	105
Troubleshooting	106
Unit Test	109
A. Solutions	113
Introduction to Red Hat Storage Server	114
Explore the Classroom Environment	116
Installing Red Hat Storage Server	120
Basic Configuration	122
Volume Types	125
Clients	129
ACLs and Quotas	136
Extending Volumes	141
IP Failover	145
Geo-Replication	149
Unified File and Object Storage	152
Troubleshooting	155

Document Conventions

Notes and Warnings



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.



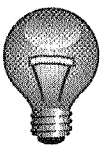
Comparison

"Comparisons" look at similarities and differences between the technology or topic being discussed and similar technologies or topics in other operating systems or environments.



References

"References" describe where to find external documentation relevant to a subject.



Important

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



Warning

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

Introduction

Welcome to class!

Thank you for attending this Red Hat training class. Please let us know if you have any special needs while at our training facility.

Please ask the instructor if you have any questions about the facility, such as operating hours of the facility and when you will have access to the classroom, locations of restrooms and break rooms, availability of telephones and network connectivity, and information about the local area.

As a courtesy to other students, please place your pager or cell phone's ringer on vibrate or mute, or turn off your devices during class. We ask that you only make calls during break periods.

If you have a personal emergency and are unable to attend or complete the class, please let us know. Thank you!

About Red Hat Enterprise Linux

This course is taught using Red Hat Enterprise Linux, an enterprise-targeted Linux distribution focused on mature open source software designed specifically for organizations using Linux in production settings.

Red Hat Enterprise Linux is sold on a subscription basis, where the subscription gives you continued access to all supported versions of the operating system in binary and source form, not just the latest one, including all updates and bug fixes. Extensive support services are included which vary depending on the subscription level selected; see <http://www.redhat.com/subscriptions/> for details. Various Service Level Agreements are available that may provide up to 24x7 coverage with a guaranteed one hour response time for Severity 1 issues. Support will be available for up to seven years after a particular major release (ten years with the optional "Extended Update Support" Add-On).

Red Hat Enterprise Linux is released on a multi-year cycle between major releases. Minor updates to major releases are released periodically during the lifecycle of the product. Systems certified on one minor update of a major release continue to be certified for future minor updates of the major release. A core set of shared libraries have APIs and ABIs which will be preserved between major releases. Many other shared libraries are provided, which have APIs and ABIs which are guaranteed within a major release (for all minor updates) but which are not guaranteed to be stable across major releases.

Red Hat Enterprise Linux is based on code developed by the open source community, which is often first packaged through the Red Hat sponsored, freely-available Fedora distribution (<http://fedoraproject.org/>). Red Hat then adds performance enhancements, intensive testing, and certification on products produced by top independent software and hardware vendors. Red Hat Enterprise Linux provides a high degree of standardization through its support for four processor architectures (32-bit Intel x86-compatible, AMD64/Intel 64 (x86-64), IBM POWER, and IBM mainframe on System z). Furthermore, we support the 4000+ ISV certifications on Red Hat Enterprise Linux whether the RHEL operating system those applications are using is running on "bare metal", in a virtual machine, as a software appliance, or in the cloud using technologies such as Amazon EC2.

Currently, the Red Hat Enterprise Linux product family includes:

- **Red Hat Enterprise Linux for Servers:** the datacenter platform for mission-critical servers running Red Hat Enterprise Linux. This product includes support for the largest x86-64 and x86-compatible servers and the highest levels of technical support, deployable on bare metal, as a guest on the major hypervisors, or in the cloud. Subscriptions are available with flexible guest entitlements of one, four, or unlimited guests per physical host. Pricing is based on the basis of the number of socket-pairs populated on the system motherboard, the number of guests supported, the level of support desired, and the length of subscription desired.

Red Hat Enterprise Linux for IBM POWER and *Red Hat Enterprise Linux for IBM System z* are similar variants intended for those system architectures.

- **Red Hat Enterprise Linux Desktop:** built for the administrator and end-user, Red Hat Enterprise Linux Desktop provides an attractive and highly productive environment for knowledge workers on desktops and laptops. Client installations can be finely tailored and locked down for simplicity and security for any workstation task.

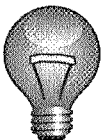
The basic *Desktop* variant is designed for task workers who have a limited amount of administrative control over the system, who primarily use productivity applications like Firefox Evolution/Thunderbird, OpenOffice.org, and Planner/TaskJuggler. The more sophisticated *Workstation* variant is designed for advanced Linux users who need a stand-alone development environment, and who are expected to have local super-user privileges or selected super-user privileges.

In addition, other variants exist such as *Red Hat Enterprise Linux for HPC Head Node* and *Red Hat Enterprise Linux for HPC Compute Node* (targeted at high-performance computing clusters), and *Red Hat Enterprise Linux for SAP Business Applications*. For more information please visit <http://www.redhat.com/>.

Additional Red Hat Enterprise Linux Software

Two additional software update channels are provided with Red Hat Enterprise Linux beyond the core software packages shipped:

- **Supplementary:** the "Supplementary" channel provides selected closed source packages, built for Red Hat Enterprise Linux as a convenience to the customer. These include things like Adobe Flash or proprietary Java JVMs.
- **Optional:** the "Optional" channel provides selected open source packages, as a convenience only. They are generally included in another Red Hat Enterprise Linux variant as a fully-supported package, or are a build requirement for the distribution. These packages are only available through a Red Hat Network child channel.



Important

Supplementary and *Optional* packages are provided with limited support, as a customer convenience only.

Red Hat also offers a portfolio of fully-supported *Add-Ons for Red Hat Enterprise Linux* which extend the features of your Red Hat Enterprise Linux subscription. These add-ons allow you to

add capabilities and tailor your computing environment to your particular needs. These Add-Ons include support for high availability application clustering, cluster file systems and very large file systems, enhanced system management with Red Hat Network, extended update support, and more.



Note

Please visit <http://www.redhat.com/rhel/add-ons/> for more information about available *Add-Ons for Red Hat Enterprise Linux*.

For information about other products which are provided by Red Hat, such as Red Hat Enterprise Virtualization, JBoss Enterprise Middleware, Red Hat Enterprise MRG, and various custom consulting and engineering services, <http://www.redhat.com/products/> also has useful information.

Contacting Red Hat Technical Support

One of the benefits of your subscription to Red Hat Enterprise Linux is access to technical support through Red Hat's customer portal at <http://access.redhat.com/>. If you do not have a Red Hat account on the customer portal or are not able to log in, you can go to <https://access.redhat.com/support/faq/LoginAssistance.html> or contact Customer Service for assistance.

You may be able to resolve your problem without formal technical support by searching Knowledgebase (<https://access.redhat.com/kb/knowledgebase/>). Otherwise, depending on your support level, Red Hat Support may be contacted through a web form or by phone. Phone numbers and business hours for different regions vary; see <https://access.redhat.com/support/contact/technicalSupport.html> for current information. Information about the support process is available at https://access.redhat.com/support/policy/support_process.html.

Some tips on preparing your bug report to most effectively engage Red Hat Support:

- **Define the problem.** Make certain that you can articulate the problem and its symptoms before you contact Red Hat. Be as specific as possible, and detail the steps you can use (if any) to reproduce the problem.
- **Gather background information.** What version of our software are you running? Are you using the latest update? What steps led to the failure? Can the problem be recreated and what steps are required? Have any recent changes been made that could have triggered the issue? Were messages or other diagnostic messages issued? What *exactly* were they (exact wording may be critical)?
- **Gather relevant diagnostic information.** Be ready to provide as much relevant information as possible; logs, core dumps, traces, the output of **sosreport**, etc. Technical Support can assist you in determining what is relevant.
- **Determine the Severity Level of your issue.** Red Hat uses a four-level scale to indicate the criticality of issues; criteria may be found at https://access.redhat.com/support/policy/GSS_severity.html.



Warning

Bugzilla is not a support tool! For support issues affecting Red Hat Enterprise Linux, customers should file their bugs through the support channels discussed above in order to ensure that Red Hat is fully aware of your issue and can respond under the terms of your Service Level Agreement. Customers should *not* file bugs directly in the <http://bugzilla.redhat.com/> web interface.

For Red Hat Enterprise Linux, Bugzilla is used by engineering to track issues and changes, and to communicate on a technical level with Engineering partners and other external parties. Anyone, even non-customers, can file issues against Bugzilla, and Red Hat does monitor them and review them for inclusion in errata.

However, Red Hat does not guarantee any SLA for bugs filed directly in Bugzilla (bypassing normal support channels). A review might happen immediately, or after a time span of any length. Issues coming through Support are always prioritized above issues of similar impact and severity filed against Bugzilla. Also, work arounds and hotfixes if possible and appropriate may be provided to customers by Support even before a permanent fix is issued through Red Hat Network.

Red Hat considers issues directly entered into Bugzilla important feedback, and it allows us to provide efficient interaction with the open source development community and as much transparency as possible to customers as issues are processed. Nevertheless, for customers encountering production issues in Red Hat Enterprise Linux, Bugzilla is not the right channel.

About This Course

Red Hat Storage Server Administration

The Red Hat Storage Server Administration course aims to teach system administrators how to setup, configure, and maintain large scale storage systems using Red Hat Storage Server.

Objectives

- Implement a scale-up/scale-out storage solution using Red Hat Storage Server.
- Create different volume types with Red Hat Storage Server.
- Implement and configure different client types for Red Hat Storage Server.
- Implement high-availability for clients using **CTDB**.

Audience and Prerequisites

- Students who are Linux system administrators with *at least* two years of full-time Linux experience, preferably using Red Hat Enterprise Linux.
- Students should enter the class with current RHCSA credentials or equivalent knowledge.

Structure of the Course

Red Hat training courses are interactive, hands-on, performance-based, real world classes meant to engage your mind and give you an opportunity to use real systems to develop practical skills. We encourage students to participate in class and ask questions in order to get the most out of their training sessions.

This course is divided up into a number of *units* organized around a particular topic area. Each unit is divided up into multiple *sections* which focus on a specific skill or task. The unit will start with an introduction to the material, then move on to the first section.

In each section, there will be a *presentation* led by the instructor. During the presentation, it may be a good idea to take notes in your student workbook (this book), and the instructor may remind you to do so. The presentation is followed by a short activity or *assessment* to give you the opportunity to practice with the material or review procedures. After a review of the assessment, the instructor will move on to the next section. At the end of the unit there will typically be a hands-on lab exercise of some sort (a "*unit test*") which will give you an opportunity to learn by doing and review your understanding of the unit's content. Please feel free to ask questions in class, or ask the instructor for advice or help during the end-of-unit exercise. We want the classroom environment to be a "safe" place where you feel comfortable asking questions and learning from things that work and things that do not work at first.

Orientation to the Classroom Network

A single network subnet is used in this course, 192.168.0.0/24, and its hosts belong to the "example.com" DNS domain. This network allows communication between the classroom server and student workstations.

Each student is assigned a physical machine (**desktopX.example.com** on 192.168.0.X) which hosts virtual machines for lab activities.

The instructor controls machines which students will see as well. The **instructor.example.com** host is the classroom utility server providing default routing services, DHCP, DNS name service, YUM software repositories used in class, and other network services.

Classroom Machines

Machine name	IP addresses	Role
desktopX.example.com	192.168.0.10×X	Physical student workstation
instructor.example.com	192.168.0.254	Physical instructor machine and utility server

Furthermore each student workstation will have five virtual machines connected to the classroom network. These machines are called **node1** through **node5** and have the following host names and IP addresses for each student workstation **desktopX** (where X is the desktop number between 1 and 20):

Virtual Machines

Node	Host name	IP address
node1	cXn1.example.com	192.168.0.X1
node2	cXn2.example.com	192.168.0.X2
node3	cXn3.example.com	192.168.0.X3
node4	cXn4.example.com	192.168.0.X4
node5	cXn5.example.com	192.168.0.X5

For example, **node5** for the student working at **desktop19** would have a host name of **c19n5.example.com** and **192.168.0.195** as an IP address. **node3** for the student at **desktop5** would be **c5n3.example.com** with IP address **192.168.0.53**.

Internationalization

Language Support

Red Hat Enterprise Linux 6 officially supports twenty-two languages: English, Assamese, Bengali, Chinese (Simplified), Chinese (Traditional), French, German, Gujarati, Hindi, Italian, Japanese, Kannada, Korean, Malayalam, Marathi, Oriya, Portuguese (Brazilian), Punjabi, Russian, Spanish, Tamil, and Telugu. Support for Maithili, Nepalese, and Sinhala are provided as Technology Previews.

System-wide Default Language

The operating system's default language is normally set to US English (en_US.UTF-8), but this can be changed during or after installation.

To use other languages, you may need to install additional package groups to provide the appropriate fonts, translations, dictionaries, and so forth. By convention, these package groups are always named **language-support**. These package groups can be selected during installation, or after installation with PackageKit (**System > Administration > Add/Remove Software**) or **yum**.

A system's default language can be changed with **system-config-language** (**System > Administration > Language**), which affects the `/etc/sysconfig/i18n` file.

Per-user Language Selection

Users may prefer to use a different language for their own desktop environment or interactive shells than is set as the system default. This is indicated to the system through the **LANG** environment variable.

This may be set automatically for the GNOME desktop environment by selecting a language from the graphical login screen by clicking on the **Language** item at the bottom left corner of the graphical login screen immediately prior to login. The user will be prompted about whether the language selected should be used just for this one login session or as a default for the user from now on. The setting is saved in the user's `~/.dmrc` file by GDM.

If a user wants to make their shell environment use the same **LANG** setting as their graphical environment even when they login through a text console or over **ssh**, they can set code similar to the following in their `~/.bashrc` file. This code will set their preferred language if one is saved in `~/.dmrc` or will use the system default if one is not:

```
i=$(grep 'Language=' ${HOME}/.dmrc | sed 's/Language=//')
if [ "$i" != "" ]; then
    export LANG=$i
fi
```

Languages with non-ASCII characters may have problems displaying in some environments. Kanji characters, for example, may not display as expected on a virtual console. Individual commands can be made to use another language by setting **LANG** on the command-line:

```
[user@host ~]$ LANG=fr_FR.UTF-8 date
lun. oct. 24 10:37:53 CDT 2011
```

Subsequent commands will revert to using the system's default language for output. The **locale** command can be used to check the current value of **LANG** and other related environment variables.

Input Methods

IBus (Intelligent Input Bus) can be used to input text in various languages under X if the appropriate language support packages are installed. You can enable IBus with the **im-chooser** command (**System > Preferences > Input Method**).

Language Codes Reference

Language Codes

Language	\$LANG value	Language package group
English (US)	en_US.UTF-8	(default)
Assamese	as_IN.UTF-8	assamese-support
Bengali	bn_IN.UTF-8	bengali-support
Chinese (Simplified)	zh_CN.UTF-8	chinese-support
Chinese (Traditional)	zh_TW.UTF-8	chinese-support
French	fr_FR.UTF-8	french-support
German	de_DE.UTF-8	german-support
Gujarati	gu_IN.UTF-8	gujarati-support
Hindi	hi_IN.UTF-8	hindi-support
Italian	it_IT.UTF-8	italian-support
Japanese	ja_JP.UTF-8	japanese-support
Kannada	kn_IN.UTF-8	kannada-support
Korean	ko_KR.UTF-8	korean-support
Malayalam	ml_IN.UTF-8	malayalam-support
Marathi	mr_IN.UTF-8	marathi-support
Oriya	or_IN.UTF-8	oriya-support
Portuguese (Brazilian)	pt_BR.UTF-8	brazilian-support
Punjabi	pa_IN.UTF-8	punjabi-support

Language	\$LANG value	Language package group
Russian	ru_RU.UTF-8	russian-support
Spanish	es_ES.UTF-8	spanish-support
Tamil	ta_IN.UTF-8	tamil-support
Telugu	te_IN.UTF-8	telugu-support
<i>Technology Previews</i>		
Maithili	mai_IN.UTF-8	maithili-support
Nepali	ne_NP.UTF-8	nepali-support
Sinhala	si_LK.UTF-8	sinhala-support



UNIT 1

INTRODUCTION TO RED HAT STORAGE SERVER

Introduction

Unit Details	
Unit Goal	Overview of Red Hat Storage Server
Unit Sections	<ul style="list-style-type: none">• Features• Use Cases• Concepts & Terminology• Hardware Requirements
Hands-On Activities	None
Unit Test	None

Features

Key Features

Red Hat Storage Server is based upon the work of the GlusterFS project to provide enterprise class storage solutions built on top of commodity hardware. Some of the more interesting features of Red Hat Storage Server are outlined below:

Scale-Up/Scale-Out

You can grow the capacity, as well as the redundancy and availability, of your Red Hat Storage pool at any time, without downtime, by adding more storage to individual servers or by adding servers to your pool.

Decentralized

Unlike other distributed storage systems, Red Hat Storage Server does not rely on a central metadata server to locate files. Instead, files are placed algorithmically across the different members of a volume, making them easy to retrieve and eliminating single-points-of-failure or I/O bottlenecks. This causes performance to scale linearly when adding extra storage servers.

High Availability

Data can be made highly-available by synchronous mirroring across multiple servers. Geo-replication can be used for asynchronous mirroring of data to a remote data center for disaster recovery purposes.

No Application Rewrites

Red Hat Storage Server provides POSIX-compliant file systems to clients, eliminating the need to rewrite applications to use storage.

Commodity Hardware

Red Hat Storage Server is designed to run on commodity x86_64 hardware. This allows you to build up an enterprise class storage network using relatively cheap components.

Bare Metal and Cloud

You can deploy Red Hat Storage Server on bare metal machines, in your private cloud, or on a public cloud like Amazon EC2. You can also combine machines across public and private clouds into a storage pool.



References

Red Hat Storage Server Administration Guide

- Chapter 1: Introduction
- Chapter 3: Key Features



Multiple Choice Quiz

Red Hat Storage Server Features

Please answer the following questions:

1. Red Hat Storage Server can be deployed to...
(select one or more of the following...)
 - a. Physical Hardware
 - b. Public Cloud
 - c. Private Cloud
 - d. Hybrid Cloud

2. Red Hat Storage Server uses a central metadata node to locate files.
(select one of the following...)
 - a. True
 - b. False

3. Red Hat Storage Server is built around the following technology:
(select one of the following...)
 - a. Ceph
 - b. DRBD
 - c. GFS2
 - d. GlusterFS

Use Cases

The flexibility, performance, and scalability of Red Hat Storage Server makes it ideally suited for a wide range of different use cases. We will highlight some of these to give you an understanding of what is possible.

Nearline and Archival Storage

Since most companies have ever-expanding storage needs, it is critical to have an easily expandable storage solution that can be accessed by a wide range of clients without the need for dedicated client software. Red Hat Storage Server can be easily deployed into a wide range of environments.

Key features for this type of workload include:

- Scalability
- Compatibility (POSIX compliant, wide range of clients)
- High Availability
- Efficient Access

High Performance Computing

High Performance Compute clusters often need very fast access to large amounts of data. Red Hat Storage Server provides an affordable solution for this type of workload.

Key features for this type of workload include:

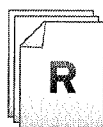
- Scalability, up to Multiple Petabytes.
- High Performance without Bottlenecks
- Infiniband Support
- Compatibility (POSIX compliant, wide range of clients)

Content Clouds

Companies distributing multimedia (and other) content to Internet customers often have to deal with hundreds of terabytes, if not petabytes, of files. Making these files available through a web front-end can provide a large challenge. Red Hat Storage Server can help make this a manageable problem.

Key features for this type of workload include:

- Scalability
- Compatibility (POSIX compliant, wide range of clients)
- High Availability
- Unified File and Object Access using *Swift*



References

Red Hat Storage Server Administration Guide

- Chapter 4: Use Case Examples

Concepts & Terminology

Trusted Storage Pool

For multiple servers to work together in a storage cluster, they have to implicitly trust each other. When you first start a Red Hat Storage Server it belongs to a *Trusted Storage Pool* with only itself as a member, but you can add servers together to form a larger pool.

Node

A server participating in a Trusted Storage Pool or cluster is sometimes referred to as a *node*. A node can only be a member of a single Trusted Storage Pool.

Brick

Red Hat Storage Server allows you to combine multiple file systems (called *Bricks*) into one larger file system called a *Volume*. When referring to a brick you can use the syntax **SERVER:MOUNTPOINT**, e.g. **storageserver1.example.com:/exports/myfirstbrick**. A single brick can only be a member of a single volume.

Metadata

Metadata is data about one or more other pieces of data. Examples of metadata include things like file names, last modification time, owner, permissions, etc...

Volume

A *volume* is a file system presented to clients. A volume consists of one or more bricks on which data will be stored. Which file goes where depends on the *Volume Type* as well as the filename of the file being stored.

Elastic Hash Algorithm

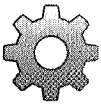
When locating a file on a volume, each Red Hat Storage Server in a pool has the intelligence to locate any piece of data without having to query an index or another server. This is done by hashing the filename of the requested data along with some other information like the number of bricks in the target volume.



References

Red Hat Storage Server Administration Guide

- Chapter 2: Red Hat Storage Architecture
- Chapter 3: Key Features



Multiple Choice Quiz

Concepts and Terminology

Please answer the following questions:

1. In order to work together, multiple Red Hat Storage Servers **must** be combined into a Trusted Storage Pool.

(select one of the following...)

- a. True
- b. False

2. When retrieving or storing a file on a distributed volume its location is determined by:

(select one of the following...)

- a. Querying a central index
- b. Querying the other nodes in the Trusted Storage Pool
- c. **/dev/urandom**
- d. An Elastic Hash Algorithm

3. A brick can belong to multiple volumes at a time.

(select one of the following...)

- a. True
- b. False

Hardware Requirements

Physical Requirements

Currently only a subset of the hardware certified for use with Red Hat Enterprise Linux is certified for use with Red Hat Storage Server, although this list is being updated on a regular basis. For an up-to-date list of certified and compatible hardware refer to the compatibility page listed in the references below. In general only 2-socket server machines are supported, with direct-attached-storage setup in hardware RAID6 arrays of 12 drives each, using 2 or 3 TB drives. Furthermore each drive/RAID controller should be battery backed or flash backed to ensure data integrity in case of a power failure.

RAM requirements differ based on the type of workload, but range from a minimum of 16 GiB for an archival workload up to a minimum of 48 GiB for an HPC workload.

For networking, a minimum of 2x10GigE network cards is recommended, but 2x1GigE is supported.

Virtual Requirements

Virtual Red Hat Storage Servers are supported on Red Hat Enterprise Virtualization 3.0 and VMware vSphere 5.x. Virtual machines should be configured with a minimum of 4 vCPUs and a minimum of 16 GiB of RAM on both hypervisors.

Public Cloud Requirements

Red Hat provides an AMI image for running Red Hat Storage on the Amazon Elastic Cloud (EC2).

Exceptions

If you are hitting any of the minimum or maximum supported configurations, or if you want to run in a setup normally not supported by Red Hat, you can request a review by Red Hat. Follow the instructions in the article linked in the references to request a review. Architecture reviews are part of your normal support agreement, but some setups might require the use of Red Hat Professional Services as well.



References

Red Hat Storage Server 2.0 Compatible Physical, Virtual Server and Client OS Platforms

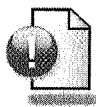
<https://access.redhat.com/knowledge/articles/66206>

Red Hat Storage Server 2.0 Exceptions & Review Process

<https://access.redhat.com/knowledge/articles/85893>



Personal Notes



Unit Summary

Features

- Scale-Up/Scale-Out
- Decentralized
- Commodity Hardware
- Bare metal and cloud

Use Cases

- Nearline and Archival Storage
- High-Performance Computing
- Content Clouds

Concepts & Terminology

- Key Concepts
- Key Terms

Hardware Requirements

- Supported Hardware
- Supported Hypervisors
- Supported Public Clouds



UNIT 2

EXPLORE THE CLASSROOM ENVIRONMENT

Introduction

Unit Details	
Unit Goal	Get familiar with the classroom environment
Unit Sections	<ul style="list-style-type: none">• Exploring the Classroom
Hands-On Activities	<ul style="list-style-type: none">• Explore the Classroom Environment
Unit Test	Accessing your Virtual Machines

Exploring the Classroom

Available Machines

During this course you will have six machines available for your use. The table below lists their hostnames, IP addresses, and primary uses.

Hostname	IP Address	Primary Use
desktopX.example.com desktopX	192.168.0.X0/24	This system can be used as a client for CIFS services and can access various services on the instructor system and the Internet (if your classroom allows Internet access). <i>Physical Classroom:</i> If you are taking this course in a physical classroom this system is a virtualization host for your other five machines.
cXn1.example.com node1/nodeX_1	192.168.0.X1/24	This is the one of four systems that you will use to configure a Red Hat Storage Server cluster.
cXn2.example.com node2/nodeX_2	192.168.0.X2/24	This is the one of four systems that you will use to configure a Red Hat Storage Server cluster.
cXn3.example.com node3/nodeX_3	192.168.0.X3/24	This is the one of four systems that you will use to configure a Red Hat Storage Server cluster.
cXn4.example.com node4/nodeX_4	192.168.0.X4/24	This is the one of four systems that you will use to configure a Red Hat Storage Server cluster.
cXn5.example.com node5/nodeX_5	192.168.0.X5/24	This system will be used as a client for your storage cluster and as a slave for Geo-Replication.

Username and Passwords

All of your systems have been configured with a **root** password of **redhat**. Your **desktopX** system also has an account called **student** with a password of **student**. It is recommended that you use this account to log into the graphical desktop on your **desktopX** system.

Instructor Systems

Your instructor manages a system called **instructor.example.com**. This system has an IP address of **192.168.0.254/24**. The instructor system performs a variety of roles in the classroom:

- DHCP
- DNS
- Routing

- Materials access over NFS/FTP/HTTP
- Presentations by your instructor

Accessing your Virtual Machines

How you access your machines depends on how you are taking this course:

If you are taking this course in a physical classroom:

After logging into your **desktopX** system as the user **student** with the password **student**, you can launch the **Virtual Machine Manager** from the main Gnome system menu by selecting **Applications > System Tools > Virtual Machine Manager**. When prompted for the **root** password enter **redhat**.

You may not see any virtual machines. They will be created by scripts that you run during your lab exercises.

Alternative method: You can also access the graphical consoles of your virtual machines using the **virt-viewer** command from the command-line shell as **root**. If you use the **-r** option, the **virt-viewer** window will remain open after the virtual machine it connects to shuts down. For example:

```
[root@desktopX ~]# virt-viewer -r node1 &
```

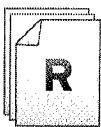
If you are taking this course in a virtual environment:

If you are taking this course in a virtual environment, an application should have launched when you connected to the classroom. Use this application to connect to your different virtual machines by double-clicking on their names in the left-hand menu. Buttons to control your virtual machines are present above the console of your virtual machine, including a **Start** and **Reset** button.

Both physical and virtual training

No matter how you take this course, you can always connect to your virtual machines using **SSH**, using the hostnames and accounts detailed above.

Using **SSH** can be especially useful when you want to run the same command on multiple machines. To do this you will probably want to configure public-key authentication using **ssh-keygen** and **ssh-copy-id**. For extra convenience you can also configure some options in **~/.ssh/config**. The unit test at the end of this unit has some examples on how to configure this.



References

Red Hat Enterprise Linux Virtualization Getting Started Guide



Performance Checklist

Explore the Classroom Environment

Lab Overview: In this exercise you will explore the classroom network and the machines available for your use.

- ☐ 1. If you haven't already done so log in to your **desktopX** machine as the user **student** with the password **student**.

If you are taking this course in a physical classroom your **desktopX** machine is the physical machine at which you are seated.

If you are taking this course in a virtual environment you can access your **desktopX** machine by locating the entry for **desktopX** in the left-hand menu and double-clicking on it to open the console. If your machine has not yet started, use the **Start** button above the console to start it.

- ☐ 2. Open a terminal on your **desktopX** machine.
- ☐ 3. Inspect the hostname for your **desktopX** machine. This should be something like **desktopX.example.com**. Note the hostname and the **X** part in the table below for future reference.
- ☐ 4. Inspect the IP address assigned to your **desktopX** machine. The address for this machine will be assigned to the **br0** interface.

Setting	Value
Hostname	
X	
IP Address	

Unit Test



Performance Checklist

Accessing your Virtual Machines

Lab Overview: In this exercise you will practice accessing your virtual machines.

- ☐ 1. Reset your **node1** virtual machines. If you are in a physical classroom this can be done by executing the command **lab-setup-nodes -1** as **root** from your **desktopX** machine. In a virtual learning environment open the console for your **node1** virtual machine and press the **Reset** button located above the console.

- ☐ 2. Open a console to your **node1/cXn1** virtual machine, and log in as **root** with the password **redhat**.

In a physical classroom you can do this by executing the command **virt-viewer node1** as **root** or by double-clicking on the **node1** virtual machine in **virt-manager** (Applications > System Tools > Virtual Machine Manager).

In a *virtual classroom* you can do this by double-clicking on the **node1/cXn1** virtual machine in the left-hand menu. This button will only be available if the machine is currently powered down, and after resetting you will have to use the **Power On** button to start your machine.

- ☐ 3. Determine the hostname of your virtual machine. This should be **cXn1.example.com**, where **X** is the number you determined in the previous exercise.
- ☐ 4. Determine the IP address for your virtual machine. This IP address should be **192.168.0.X1**, where **X** is your desktop number, and it should be assigned to the **eth0** interface.
- ☐ 5. Open a terminal as the user **student** on your **desktopX** machine and create a new **SSH key-pair**. Do not set a passphrase on this key-pair for classroom use.
- ☐ 6. Copy the public part of your newly generated key-pair to the **root** account on your **cXn1** machine so you can use **SSH** without entering a password.
- ☐ 7. To make your life easier, you can configure **SSH** to always connect to the **root** account when you **ssh** into your virtual machines. To do this create a new file **/home/student/.ssh/config** on **desktopX** with the following content. Make sure to replace **X** with your desktop number.

```
Host cXn* cXn*.example.com
  User root
```

Set the permissions on that file to **0600**.

- ☐ 8. As **student** on your **desktopX** machine, attempt to **ssh** to **cXn1.example.com**. This should log you in automatically as **root**.
- ☐ 9. Determine the default runlevel of your **cXn1** machine.

- 10. Determine which services are started in the default runlevel.



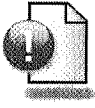
Note

If you want to use password-less login with **SSH** key-pairs, copy your public key to every machine you want to access in this manner.

When you reset a virtual machine it will be brought back to a pristine state, and as such you will have to copy your keys again.



Personal Notes



Unit Summary

Exploring the Classroom

- Explore the machines available to you
- Access your virtual machines using both a console and **ssh**.



UNIT 3

INSTALLING RED HAT STORAGE SERVER

Introduction

Unit Details	
Unit Goal	Install Red Hat Storage Server
Unit Sections	<ul style="list-style-type: none">• Installing Red Hat Storage Server on Premise• Installing Red Hat Storage Server on Public Cloud
Hands-On Activities	<ul style="list-style-type: none">• Installing Red Hat Storage Server
Unit Test	None

Installing Red Hat Storage Server on Premise

Red Hat Storage Server for On-Premise can be installed from a DVD image you can download from the Red Hat Customer Portal (<http://access.redhat.com>). You can burn this image to a DVD, or you can make it available using a PXE environment.

Installing Red Hat Storage Server is very similar to installing Red Hat Enterprise Linux, but with far fewer choices to make during the installation process. The basic procedure is outlined below.

1. Obtain an installation ISO for Red Hat Storage Server from <https://access.redhat.com>.
2. Make this installation medium available for installations. Burn it to a recordable DVD or upload it to the virtual DVD drive of a management card for your server or make it available via PXE boot.
3. Boot the server you want to install from the prepared installation medium. You will be greeted by a screen like the one below. Press **Enter** or wait a couple of seconds to continue to the installer.

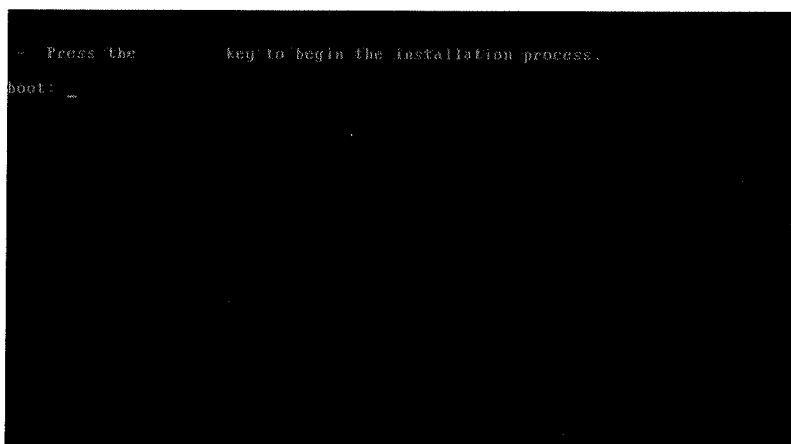


Figure 3.1: The boot screen of the Red Hat Storage Server Installer

4. The first screen you encounter will ask for the **root** password. Enter the same password twice then click on **Next** to continue.

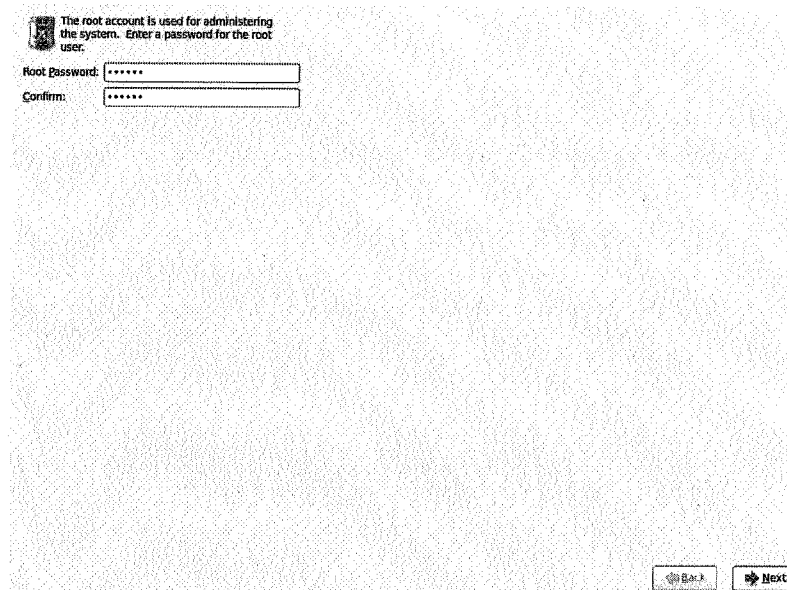


Figure 3.2: The password configuration screen

5. On the partitioning screen select one of the layout types to use. It is *highly* recommended to check the **Review and modify partitioning layout** checkbox since we want to make sure that we do not use our large storage arrays for the operating system itself.

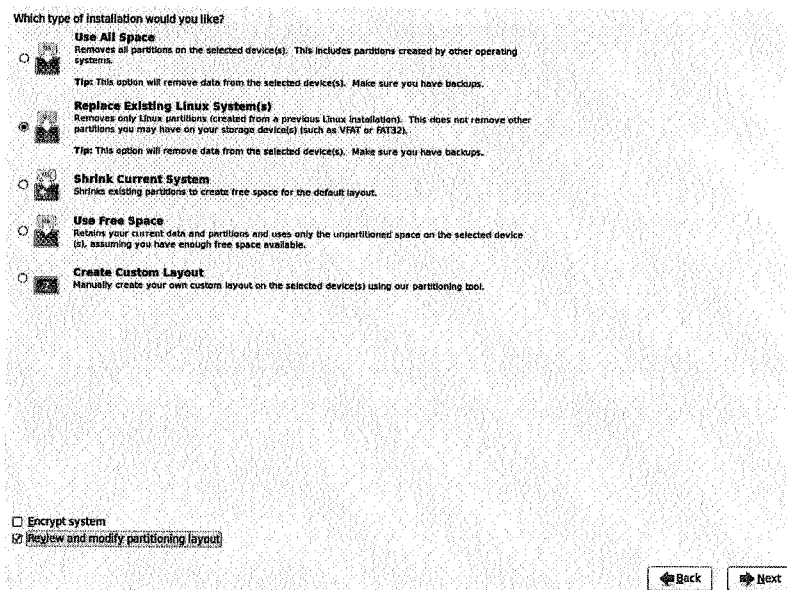


Figure 3.3: The partitioning configuration screen

6. On the advanced partitioning screen make sure you have allocated enough space for the root file system (≥ 2048 MiB). Also make sure you are not using the storage arrays you want to use for the storage of your bricks for the operating system itself. Click on **Next** when you are ready to start the installation.

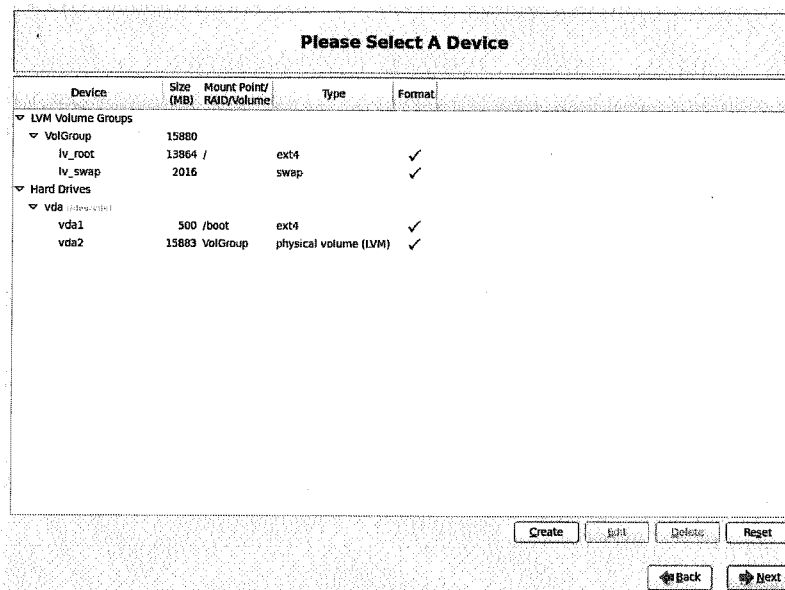


Figure 3.4: The advanced partitioning configuration screen

- Before the installation starts you might be asked to confirm that you really want to format your drives. Once you have confirmed this the installer will create your file systems then proceed to install the default package set for Red Hat Storage Server.

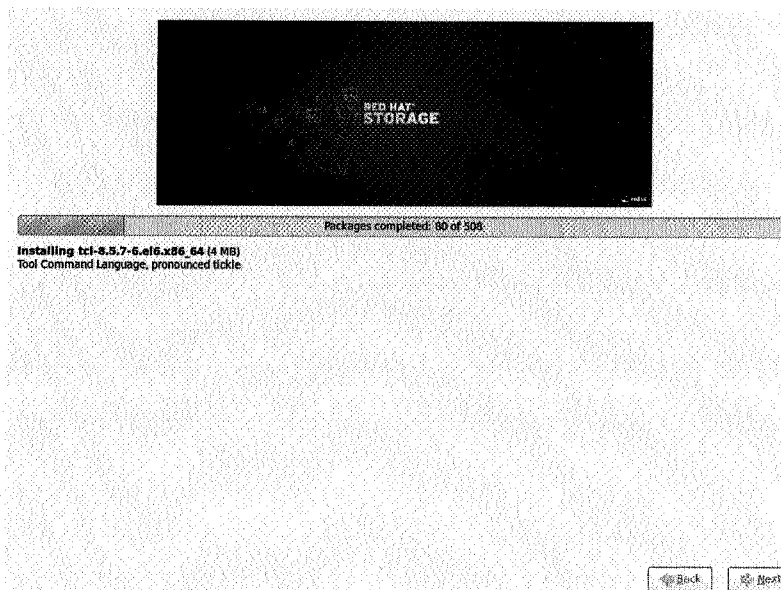


Figure 3.5: Red Hat Storage Server installation in progress

- Once the installer finishes you will be asked to click on the **Reboot** button to finish the installation.

Once your machine reboots, log in as **root** using the password you specified earlier.

9. Run the **rhn_register** command to register your Storage Server with Red Hat Network.

```
[root@storageserver ~]# rhn_register
```

Follow the on-screen instructions to register. Make sure to select **Limited Updates Only** and **RHEL EUS Server (v. 6.2.z for 64-bit x86_64)**.

10. Log into the **Classic Management** interface at <https://rhn.redhat.com> and locate your newly registered system.
11. Navigate to **Software > Software Channels** for your system and confirm the base channel is set to **RHEL EUS (v. 6.2.z for 64-bit x86_64)**.
12. Expand **Release Channels for Red Hat Enterprise Linux 6.2 for x86_64** and select **RHEL EUS Server Scalable File System (v. 6.2.z for x86_64)**.
13. Expand **Additional Services Channels for Red Hat Enterprise Linux 6.2 for x86_64** and select **Red Hat Storage Server 2.0 (RHEL 6.2.z for x86_64)**.
14. Click on the **Change Subscriptions** button to update your selection.



References

Red Hat Storage Server Installation Guide



Performance Checklist

Installing Red Hat Storage Server

Lab Overview: In this exercise you will install a new virtual machine with Red Hat Storage Server.

Before you begin...

Some instructions in this exercise differ based on whether you are taking this course in a physical classroom or in a virtual environment. Pay close attention to the instructions in the different steps to see if those steps apply to your environment.

- ❑ 1. Reset your **node1** virtual machine, then when the machine is starting press **Ctrl+B** when you see the **gPXE** prompt.

At the **gPXE** prompt type **autoboot** to start a PXE installation.

Sometime **autoboot** fails the first time. If this happens to you try again.

Physical Classroom Only: To reset your **node1** virtual machine run the following command as root on your **desktopX** machine:

```
[root@desktopX ~]# lab-setup-nodes -1
```

If this is the first time that you are resetting a virtual a base image will be downloaded from the instructor server.

Virtual Classroom Only: Power down your **node1** virtual machine, then power it back on. When you see the **gPXE** prompt press **Ctrl+B**, then type **autoboot**. You will have to click inside your console window first before keypresses are registered. Do *not* use the **install** button, as it will simply power on your machine.

- ❑ 2. When the PXE boot menu appears use your cursor keys to select **Red Hat Storage Server**, then press **Enter**.

- ❑ 3. Enter **redhat** in both the **Root Password** and **Confirm** fields, then click on **Next**.

Physical classroom only: If you do not see the **Next** button select **View > Resize to VM**.

- ❑ 4. Click on the **Use Anyway** button when you are informed of your weak password choice.
- ❑ 5. Select **Use All Space** and **Review and modify partitioning layout**, then click on **Next**.
- ❑ 6. Inspect the default partitioning selected for you, then click on **Next**.

Note: If you are performing an installation outside of the classroom you will want to pay close attention to this screen. Make sure the installer is not allocating the drives/space that you intended for your bricks to the root file system.

- ❑ 7. Accept that your drives will be wiped irrevocably by clicking on the **Format** button.
- ❑ 8. Sit back, relax, and enjoy the beautiful progress bar that now graces your screen. This installation should only take a few moments.

- 9. Once the installer has finished, click the **Reboot** button and wait for the system to shut down. (**Virtual Machine Manager** does not reboot a machine in installation mode, instead it shuts it down.) Click the **Power on the virtual machine** button (shaped like a *Play* icon).
- 10. Wait for the machine to start then login as **root** using the password you specified at the beginning of the installation process (**redhat**).
- 11. Verify the **glusterd** service has been automatically started.

```
[root@localhost ~]# service glusterd status  
glusterd (pid 1353) is running...
```

- 12. Shut down your virtual machine.

```
[root@localhost ~]# poweroff
```

Installing Red Hat Storage Server on Public Cloud

Installation

Installing Red Hat Storage Server on the Amazon EC2 public cloud is slightly different from installing Red Hat Storage Server on-premise. Instead of going through an installation process you start with a pre-existing image, also called an *AMI* (Amazon Machine Image).

Walk through the normal steps for provisioning a machine on EC2 as you normally would, but make sure you select a *Large* instance and the Red Hat Storage Server AMI.

Provisioning Storage

Since I/O performance on regular EBS volumes can be inconsistent Red Hat recommends configuring RAID0 stripes consisting of 8 equal sized EBS volumes for your storage bricks. Refer to the Red Hat Storage Server Administration Guide available on <https://access.redhat.com> for more information on how to accomplish this.

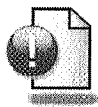


References

Red Hat Storage Server Installation Guide



Personal Notes



Unit Summary

Installing Red Hat Storage Server on Premise

- Install Red Hat Storage Server

Installing Red Hat Storage Server on Public Cloud

- Install Red Hat Storage Server on EC2



UNIT 4

BASIC CONFIGURATION

Introduction

Unit Details	
Unit Goal	Build your first Red Hat Storage Server volume
Unit Sections	<ul style="list-style-type: none">• Building a Trusted Storage Pool
Hands-On Activities	<ul style="list-style-type: none">• Build a Volume
Unit Test	None

Building a Trusted Storage Pool

Preparing Nodes

In order to achieve maximum performance you should apply some tuning settings to your nodes. One of the easiest ways to achieve this is to use the **rhs-high-throughput** tuned profile.

To apply this profile issue the following command:

```
[root@cXnY ~]# tuned-adm profile rhs-high-throughput
```

Configure a Trusted Storage Pool

Once your servers are installed, link them together to form a *Trusted Storage Pool*. This can be accomplished with the **gluster peer probe <server>** command.



Note

Almost every action you will perform on your trusted storage pool and volumes is done with the **gluster** command. The basic syntax for the **gluster** command is **gluster <object> <action> [options and arguments]....**

Run **gluster help** to get an overview of all objects and actions or run **gluster <object> help** to get help on the actions you can perform on a specific object type.

To remove a node from your trusted storage pool you can use the command **gluster peer detach <server>**.



Demonstration

Configure a Trusted Storage Pool

In this demonstration your instructor will demonstrate how to configure a trusted storage pool.

1. Log into one of the machines you wish to use as part of your trusted storage pool and run the following command:

```
[root@c0n1 ~]# gluster peer status
No peers present
```

This output indicates we have not added any trusted peers yet.

2. Execute the **gluster peer probe <server>** command to add another server to your trusted storage pool.

```
[root@c0n1 ~]# gluster peer probe c0n2.example.com
Probe successful
```

3. Run **gluster peer status** again to verify the previous action was successful.

```
[root@c0n1 ~]# gluster peer status
Number of Peers: 1

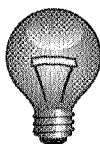
Hostname: c0n2.example.com
Uuid: f02c932d-3115-473f-bf26-89dc89490d3f
State: Peer in Cluster (Connected)
```

Creating Bricks

In order to start building *Volumes* we will first need to create *Bricks*. A Brick is an XFS file system (with 512 byte inodes) mounted on one of your Storage Servers.

It is recommended that you give your bricks a descriptive mount-point to easily identify them. Common locations for bricks include sub-directories under **/exports** and **/bricks**, e.g. **/exports/largedata-brick-A** or **/bricks/brick1**. Your mountpoints should also be unique across your trusted storage pool.

On a production system your physical volumes should be created on top of RAID6 arrays consisting of 12 drives. You should set the RAID6 stripe size to match your average file size for optimal performance.



Important

Do not forget to set the *inode size* to 512 bytes when creating your XFS file systems. Red Hat Storage Server stores metadata in the *extended attributes* area of your inodes and the default size of 256 bytes is not enough in most cases. If you are planning to use Unified File and Object Storage you should set your inode size to 1024 bytes.



Demonstration

Configuring Bricks

In this demonstration your instructor will demonstrate how to create bricks.

1. Create a *logical volume* of the desired size.

```
[root@c0n1 ~]# lvcreate -L 2G -n brick1 vg_bricks
Logical volume "brick1" created
```

2. Use the **mkfs.xfs** command to create an XFS file system on your new logical volume. Remember to specify the **-i size=<size>** option to set your inode size to 512 bytes.

```
[root@c0n1 ~]# mkfs.xfs -i size=512 /dev/vg_bricks/brick1
meta-data=/dev/vg_bricks/brick1 isize=512    agcount=4, agsize=131072 blks
=                               sectsz=512   attr=2
data      =                       bsize=4096   blocks=524288, imaxpct=25
=                               sunit=0      swidth=0 blks
naming    =version 2              bsize=4096   ascii-ci=0
log       =internal log          bsize=4096   blocks=2560, version=2
=                               sectsz=512   sunit=0 blks, lazy-count=1
realtime  =none                  extsz=4096   blocks=0, rtextents=0
```

3. Create a mount-point for your new brick.

```
[root@c0n1 ~]# mkdir -p /bricks/brick1
```

4. Create an **/etc/fstab** entry to persistently mount your brick on the mount point you just created.

```
/dev/vg_bricks/brick1 /bricks/brick1 xfs defaults 0 2
```

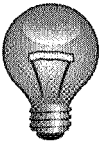
5. Mount your new brick.

```
[root@c0n1 ~]# mount /bricks/brick1
```

Building a Volume

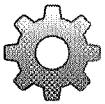
Now that we have a trusted storage pool and some bricks, we can build our first volume. You can build a volume using the **gluster volume create <VOLUME-NAME> [stripe and replica options] <BRICK>...** command.

Your volume will be *distributed* if you do not specify any *stripe* or *replica* options. The different volume types are described in detail in the next unit.



Important

New volumes are *not* automatically started after they are created, so remember to start your volumes if you want to access them.



Demonstration

Building a Volume

In this demonstration your instructor will demonstrate how to build a basic distributed volume.

1. Request a list of all volumes present on your trusted storage pool

```
[root@c0n1 ~]# gluster volume list
No volumes present in cluster
```

2. Use the **gluster volume create** command to create a new volume.

```
[root@c0n1 ~]# gluster volume create demovol \
> c0n1.example.com:/bricks/brick1 \
> c0n2.exmaple.com:/bricks/brick2
Creation of volume demovol has been successful. Please start the volume to access data.
```

3. Request another list of volumes on your cluster to confirm your new volume has been created.

```
[root@c0n1 ~]# gluster volume list
```

```
demovol
```

- Request detailed information about your new volume.

```
[root@c0n1 ~]# gluster volume info demovol

Volume Name: demovol
Type: Distribute
Volume ID: 014e2d37-c8ef-4193-aa95-4f27bd4eb9d8
Status: Created
Number of Bricks: 2
Transport-type: tcp
Bricks:
Brick1: c0n1.example.com:/bricks/brick1
Brick2: c0n2.example.com:/bricks/brick2
```

- Notice how the **Status** field lists **Created**? This means the volume has not been started yet. Let's remedy that.

```
[root@c0n1 ~]# gluster volume start demovol
Starting volume demovol has been successful
```

- Inspect your volume again to verify that it has been started.

```
[root@c0n1 ~]# gluster volume info demovol

Volume Name: demovol
Type: Distribute
Volume ID: 014e2d37-c8ef-4193-aa95-4f27bd4eb9d8
Status: Started
Number of Bricks: 2
Transport-type: tcp
Bricks:
Brick1: c0n1.example.com:/bricks/brick1
Brick2: c0n2.example.com:/bricks/brick2
```

- Request some detailed status information from your volume.

```
[root@c0n1 ~]# gluster volume status demovol
Status of volume: demovol
Gluster process                                Port      Online   Pid
-----
Brick c0n1.example.com:/bricks/brick1          24009     Y        2864
Brick c0n2.example.com:/bricks/brick2          24009     Y        2781
NFS Server on localhost                        38467     Y        2870
NFS Server on c0n2.example.com                 38467     Y        2787
```



References

- Red Hat Storage Server Administration Guide
 - Chapter 7: Setting up Trusted Storage Pools
 - Chapter 8: Setting up Red Hat Storage Volumes



Performance Checklist

Build a Volume

Lab Overview: In this exercise you will create your first Red Hat Storage Server cluster and your first volume.

Success Criteria: You will have successfully completed this lab when you have a new, running volume called **firstvol**.

Lab Outline In this exercise you will build a two-node Red Hat Storage Server cluster, then create storage bricks and a storage volume.

- ☐ 1. Reset your **node1** and **node2** virtual machines.

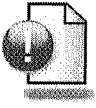
In a physical classroom, this can be done by executing the **lab-setup-nodes -12** command as **root** from your **desktopX** machine.

In a virtual learning environment, open the consoles and press the **Reset** button above the consoles for your **node1** and **node2** virtual machines. This button will only be available if the machines are currently powered down, and you will have to manually power on those machine using the **Power On** button afterwards.

- ☐ 2. Log into your **cXn1.example.com** system as **root** and peer with your **cXn2.example.com** system. Remember that **X** is your desktop number.
- ☐ 3. Verify the systems have successfully peered from your **cXn1** system.
- ☐ 4. On your **cXn1** system create a new 2 GiB logical volume in the **vg_bricks** volume group called **brick1**. The **vg_bricks** volume group has already been created for you.
- ☐ 5. On your **cXn2** system create a new 2 GiB logical volume in the **vg_bricks** volume group called **brick2**. The **vg_bricks** volume group has already been created for you.
- ☐ 6. Create an XFS file system on both of your new logical volumes. Be sure you set the inode size to 512 bytes.
- ☐ 7. On your **cXn1** and **cXn2** machines respectively, create the **/bricks/brick1** and **/bricks/brick2** mount points.
- ☐ 8. On both your nodes *persistently* mount your new file systems on the mount points you just created.
- ☐ 9. From your **cXn1** machine create a new volume called **firstvol**, using the two bricks you have just created. Do not specify any other options.
- ☐ 10. From your **cXn1** machine start your new volume.



Personal Notes



Unit Summary

Building a Trusted Storage Pool

- Configure a Trusted Storage Pool
- Create Bricks
- Create and Start a Volume



UNIT 5

VOLUME TYPES

Introduction

Unit Details	
Unit Goal	Understand the different volume types
Unit Sections	<ul style="list-style-type: none">• Different Volume Types
Hands-On Activities	<ul style="list-style-type: none">• Create a Replicated Volume
Unit Test	Creating a Distributed-Replicated Volume

Different Volume Types

There are different ways to combine bricks into a specific type of volume, and you can combine these volume types in different ways as well. This allows you to configure volumes with the performance and redundancy characteristics to match almost any workload scenario.

Distributed Volumes

Distributed volumes are the default volume type that will be created when you don't specify any options when creating a volume. In a distributed volume any file always lives on one brick. Which brick the file will end up on is based on the *elastic hash algorithm*.

This way of distributing files allows for maximum usage of available storage resources, but when a brick becomes unavailable you lose access to all of the files stored on that brick until the brick becomes available again.

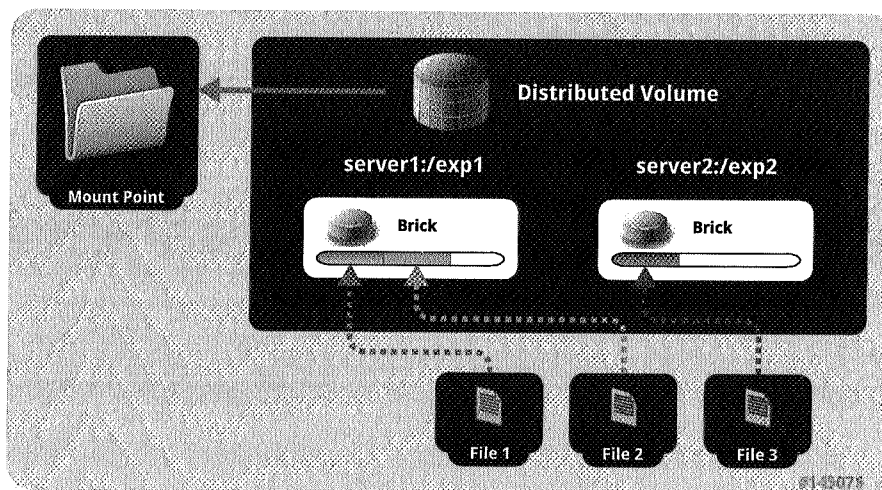


Figure 5.1: Distributed Volume

Replicated Volumes

In a *Replicated* volume, bricks are mirrored to each other. This means that a file that is written to one brick will also be written to one or more other bricks. A replicated volume is created by specifying the **replica <number of replicas>** option when creating or extending a volume, for example:

```
[root@server ~]# gluster volume create replicavolume replica 2 <BRICKS>...
```

Replicated volumes offer you redundancy and high-availability, but at the cost of reduced storage capacity.

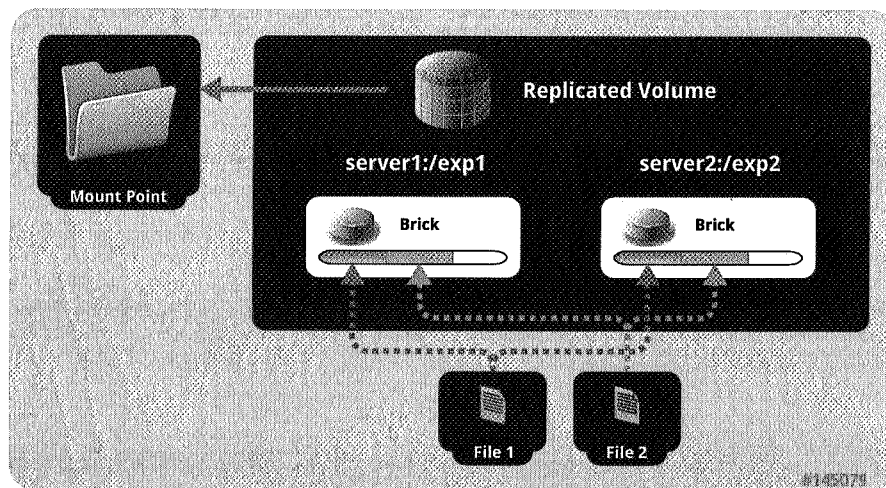


Figure 5.2: Replicated Volume



Note

When creating a replicated volume the number of bricks you specify must be a multiple of the number of replicas you request. If you specify more bricks than replicas you will get a *Distributed-Replicated* volume. This type of volume is discussed later on in this section.

Striped Volumes

In a *Striped* volume, larger files are chopped up into chunks which are then distributed across the bricks. The default stripe size is 128 KiB, but this can be configured to suit your performance needs. Striped volumes are not common, but may be of use if you have to deal with very large files on your volume.

Specify the **stripe <number of stripes>** option to create a striped volume. The number of bricks specified must be a multiple of the stripe count, for example:

```
[root@server ~]# gluster volume create stripevolume stripe 3 <BRICKS>...
```

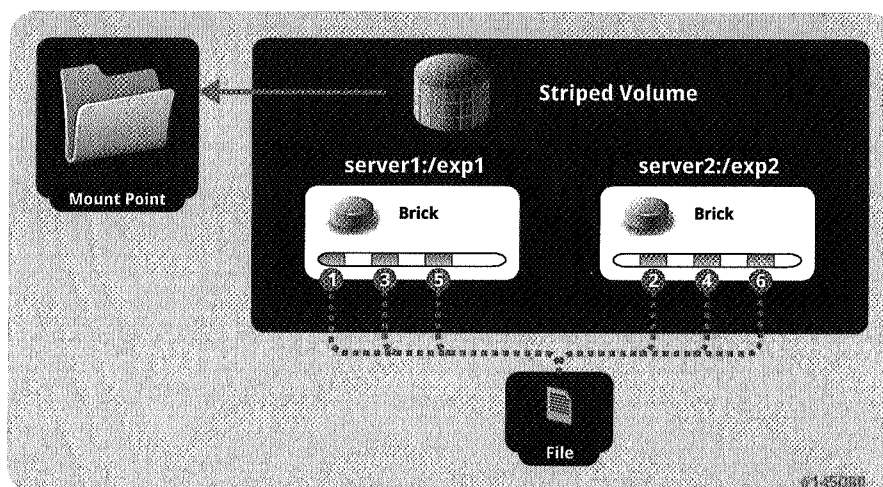


Figure 5.3: Striped Volume



Important

Striped volumes are only useful for multi-gigabyte files. Striped volumes are currently in *Technology Preview* and should not be used for production.



Note

The stripe size for a volume can be controlled by setting the `cluster.stripe-block-size` property on a volume. Setting volume properties will be discussed in a later unit.

Combined Volume Types

You can combine volume types if you want. This allows you to create volume types such as **Distributed-Replicated**, **Distributed-Striped**, **Striped-Replicated**, and **Distributed-Striped-Replicated**.

When creating a combined volume, make sure you specify a number of bricks that is a multiple of your stripe count times your replica count. For example, when you specify 2 stripes and 3 replicas you must specify a number of bricks that is a multiple of 6 (2×3). This means 6 bricks, or 12 bricks, etc.

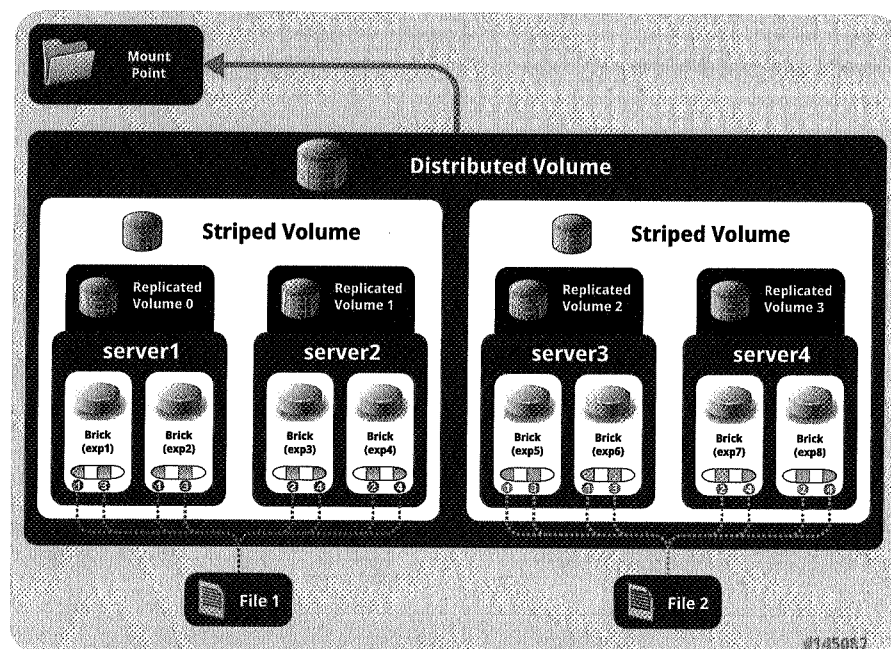


Figure 5.4: Distributed-Striped-Replicated Volume

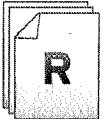
As you can see in the image above your bricks will always be first subdivided into mirrors (if **replica > 1**), then those mirrors will be sub-divided into stripes (if **stripe > 1**), and finally your files will be distributed across those stripes.



Note

Distributed-replicated volumes and striped-replicated volumes are sometimes referred to as $X \times Y$ volumes. In these cases X is the distribution or stripe count, and Y is the replica count.

Distributed-Striped-Replicated volumes are also referred to as $X \times Y \times Z$ volumes. In these cases X is the distributed count, Y is the stripe count, and Z is the replica count.



References

Red Hat Storage Server Administration Guide

- Chapter 8: Setting up Red Hat Storage Volumes



Performance Checklist

Create a Replicated Volume

Lab Overview: In this exercise you will create a replicated volume using two 2 GiB bricks located on **cXn3** and **cXn4**.

Success Criteria: You will have successfully completed this lab when you have a new, running volume called **replvol** consisting of two bricks.

Lab Outline In this exercise you will add two more nodes to your cluster, add bricks to your new nodes, then create a new replicated volume.

Before you begin...

Make sure you still have a working two-node cluster on your **cXn1** and **cXn2** machines.

- ☐ 1. Begin by resetting your **node3** and **node4** virtual machines.

Students in a physical classroom can do this by executing the **lab-setup-nodes -34** command as **root** on their **desktopX** machine.

Students in a virtual training environment must open the consoles for their **node3** and **node4** machines and click the **Reset** button above those consoles. Remember that this button will only be available when a machine is currently powered off, and you will have to manually start those machines using the **Power On** button afterwards.
- ☐ 2. From your **cXn1** machine, add your **cXn3.example.com** and **cXn4.example.com** machines to your cluster.
- ☐ 3. Confirm the nodes have been added to your cluster.
- ☐ 4. On both of your new nodes create a 2 GiB logical volume in the **vg_bricks** volume group called **brickY**, where **Y** is the node number. Format it with an XFS file system with 512-byte inodes and persistently mount it on **/bricks/brickY**.
- ☐ 5. Using the two new bricks you just created, create a new *Replicated* volume called **replvol**.
- ☐ 6. Inspect the properties of your new volume.
- ☐ 7. Start your new volume.

Unit Test



Case Study

Creating a Distributed-Replicated Volume

Lab Overview: In this exercise you will create a distributed-replicated volume using four bricks.

Success Criteria: You have successfully completed this lab when you have a new, running distributed-replicated volume called **distreplvol** using four bricks.

Lab Outline In this lab you will create the bricks necessary for a four brick distributed-replicated volume, then create and start that volume.

Before you begin...

Make sure you still have a working four-node cluster.

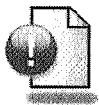
You have been asked to create a new **2 × 2 distributed-replicated** volume using four 2 GiB bricks. The volume should be called **distreplvol**.

The four bricks should be mounted on **/bricks/brick5** through **/bricks/brick8**, and their logical volume names should be **/dev/vg_bricks/brick5** through **/dev/vg_bricks/brick8** respectively. The bricks should be distributed in such a way that **brick5** resides on **cXn1.example.com**, **brick6** on **cXn2.example.com**, etc.

How would you address the case study described above? Take notes on your process in the space below and then implement it.



Personal Notes



Unit Summary

Different Volume Types

- Explain different Volume Types
- Configure different Volume Types



UNIT 6

CLIENTS

Introduction

Unit Details	
Unit Goal	Access data on Red Hat Storage volumes
Unit Sections	<ul style="list-style-type: none">• Native Client• NFS Clients• CIFS Clients• Volume Options
Hands-On Activities	<ul style="list-style-type: none">• Using the Native Client• Using NFS Clients• Configuring CIFS Clients• Setting Volume Options
Unit Test	Limit Access to Volumes

Native Client

The recommended way to access your Red Hat Storage Server volumes is to use the *Native Client*. The native client is built around the *FUSE (File system in USErspace)* technology. The native client supports using POSIX ACLs (covered in a later unit) and automatic failover.

Installing the Native Client

The native client is already installed by default on Red Hat Storage Servers. For x86_64 based Red Hat Enterprise Linux 5 and Red Hat Enterprise Linux 6 systems, you can install the native client by subscribing the machine to the **Red Hat Storage Native Client** channel. This channel can be found on the subscriptions page on Red Hat Network for your machine under the **Additional Services Channels for Red Hat Enterprise Linux 5 | 6 for x86_64**.

After subscribing your machine to this channel, install the *glusterfs-fuse* package using **yum**:

```
[root@server ~]# yum install glusterfs-fuse
```

Using the Native Client

Mount Red Hat Storage volumes by hand using the **mount** command or automatically from **/etc/fstab**. Specify **glusterfs** as the filesystem type and refer to your volume as **<storage-server>:<volume>**, where **<storage-server>** is one of the storage servers in your trusted storage pool and **<volume>** is the name of the volume you wish to mount.



Important

The native client communicates directly with the different bricks in your volume, therefore it is important that the server hostnames and IP addresses you use when adding servers to your storage pool, as well as those used when adding bricks, are resolvable and reachable by your clients.

There are a number of mount options that can be specified when mounting a volume using the native client. The table below highlights some of the more common ones.

Option	Use
backupvolfile-server=<server>	In case the server specified in the volume address is unreachable, fall back to using the server specified in this option to retrieve information about the volume.
ro	Mount the volume read-only.
acl	Enable POSIX ACLs on this volume. Using the native client ACLs can be both enforced and set. POSIX ACLs are covered in a later unit.
selinux	Enable SELinux extended attributes on this volume.

Example: Manually mount a volume using POSIX ACLs

```
[root@server ~]# mount -t glusterfs -o acl storage-server:/volume /mnt/volume
```

Example: Automatically mount a volume using **/etc/fstab** and a backup volfile-server

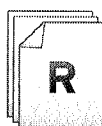
```
storage-server-1:/volume /mnt/volume glusterfs _netdev,backupvolfile-server=storage-  
server2 0 0
```



Important

Specify the **_netdev** mount option in **/etc/fstab** since a working network connection is needed to access our volume. Without this option the system startup scripts would attempt to mount this file system before networking was started, resulting in a failure.

Normally the **_netdev** mount option in **/etc/fstab** is handled by the **rc.sysinit** and **netfs** init scripts. If you use a **mount -a** command from the command line the **glusterfs-fuse** mount will complain about an unknown mount option, you can safely ignore this error.



References

How to Install Native Client RPMs for Red Hat Storage on a RHEL Server
<https://access.redhat.com/knowledge/articles/145553>

Red Hat Storage Server Administration Guide
• Section 9.1: Native Client



Performance Checklist

Using the Native Client

Lab Overview: In this exercise you will configure your **cXn5.example.com** machine to persistently mount your **firstvol** volume on **/mnt/firstvol** using the native client.

Success Criteria: You will have successfully completed this lab when you have mounted your **firstvol** volume using the native client.

Before you begin...

Make sure you still have a running **firstvol** volume on your cluster.

- ☐ 1. Reset your **node5** virtual machine.

In a physical classroom this can be done by executing the **lab-setup-nodes -5** command as **root** from the desktopX machine.

In a virtual learning environment, open the console for the **node5** virtual machine and click the **Reset** button located above the console. This button will only be available if the machine is currently powered down, and you have to manually start the machine using the **Power On** button afterwards.

- ☐ 2. On your **cXn5.example.com** machine create a mount point called **/mnt/firstvol**.
- ☐ 3. On your **cXn5.example.com** machine add an **/etc/fstab** entry to automatically mount the **firstvol** volume on **/mnt/firstvol** using the native client. Remember to add the **_netdev** mount option to make sure the mount will work at boot time.
- ☐ 4. Activate your new mount on **cXn5**.
- ☐ 5. On **cXn5** create 10 new files called **file1** through **file10** in **/mnt/firstvol**.
- ☐ 6. On **cXn1** and **cXn2** inspect the contents of the bricks that make up your **firstvol** volume (**/bricks/brick1** and **/bricks/brick2** respectively).

What do you notice about the file distribution?

NFS Clients

By default Red Hat Storage Server volumes are reachable over NFSv3 using TCP, although this behavior can be disabled per volume. NFS clients are available on a wide range of platforms and can be useful for those platform where the native client is not available.

To mount a volume using NFSv3 specify the file system type as **nfs** and use **<storage-server>:<volume>** as the source.



Important

Red Hat Enterprise Linux 6 clients will automatically attempt to mount NFS shares using NFSv4. To override this behavior, specify the **vers=3** mount option when mounting Red Hat Storage volumes using NFS.

Example: Manually mounting a Red Hat Storage volume using NFSv3 from Red Hat Enterprise Linux 6:

```
[root@server ~]# mount -t nfs -o vers=3 storage-server:/volume /mnt/volume
```

Example: Automatically mounting a Red Hat Storage volume on Red Hat Enterprise Linux 6 using **/etc/fstab**:

```
storage-server:/volume /mnt/volume nfs vers=3 0 0
```



Note

For clients that default to using UDP for NFS mounts, specify the mount option **proto=tcp** to force the client to use TCP.



Important

NFS clients do *not* automatically fail over to a different host if the host they initiated the mount with fails. This behavior this can be configured using **ctdb** and a floating IP address. We will describe how to configure this in a later unit.



References

Red Hat Storage Server Administration Guide

- Section 9.2: NFS



Performance Checklist

Using NFS Clients

Lab Overview: In this exercise you will mount your **replvol** volume over NFSv3 using TCP.

Success Criteria: You will have successfully completed this lab when your **replvol** volume is persistently mounted on **/mnt/replvol** on **cXn5** using NFS.

Before you begin...

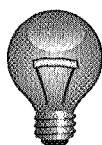
Make sure your **cXn5** machine is still operational and your **replvol** volume is still running.

- ☐ 1. On your **cXn5.example.com** machine create a mount point called **/mnt/replvol**.
- ☐ 2. On your **cXn5.example.com** machine create an **/etc/fstab** entry to automatically mount your **replvol** volume on **/mnt/replvol** using NFSv3.
- ☐ 3. Activate your new mount on **cXn5**.
- ☐ 4. On **cXn5** create 10 new files called **file1** through **file10** in **/mnt/replvol**.
- ☐ 5. On **cXn3** and **cXn4** inspect the contents of the bricks that make up your **replvol** volume (**/bricks/brick3** and **/bricks/brick4** respectively).

What do you notice about the file distribution?

CIFS Clients

On a default installation Red Hat Storage Server will automatically configure CIFS exports using **samba** for any volume that is started. This behavior is controlled using the two *hook* scripts `/var/lib/glusterd/hooks/1/start/post/S30samba-start.sh` and `/var/lib/glusterd/hooks/1/stop/pre/S30samba-stop.sh`. The names for the CIFS shares that are automatically generated in this way are **gluster-*<VOLUME NAME>***.



Important

Even though Red Hat Storage Server will automatically restart **samba** for you when you start a volume, this does not happen automatically when your server reboots or when the **samba** service is stopped. Therefore it is recommended to manually enable the **smb** service on all your nodes if you are planning on using CIFS clients.

```
[root@server ~]# chkconfig smb on
```

Samba user accounts are not automatically configured for you, so you will still have to manually configure **samba** in `/etc/samba/smb.conf` to bind to an authentication provider such as an Active Directory server or configure user accounts manually using **smbpasswd**.



Demonstration

Configuring CIFS clients

In this demonstration your instructor will demonstrate how to configure CIFS clients.

1. Create a new samba-only user **cifsuser**

```
[root@c0n1 ~]# useradd -s /sbin/nologin cifsuser
[root@c0n1 ~]# smbpasswd -a cifsuser
New SMB password: redhat
Retype new SMB password: redhat
Added user cifsuser.
```

2. Make sure the **smb** service is enabled and running.

```
[root@c0n1 ~]# chkconfig smb on
[root@c0n1 ~]# service smb start
```

3. Mount the desired volume.

```
[root@instructor ~]# mount -t cifs -o user=cifsuser,pass=redhat //c0n1.example.com/
gluster-demovol /mnt
```

4. Verify the file system was mounted successfully.

```
[root@instructor ~]# df -h /mnt
Filesystem      Size  Used Avail Use% Mounted on
```

```
//c0n1.example.com/gluster-demovol
4.0G 65M 4.0G 2% /mnt
```

5. Unmount the volume.

```
[root@instructor ~]# umount /mnt
```



Note

In this setup CIFS clients will not automatically fail over if the original host becomes unreachable. You will also have to create the samba users on all of your nodes. Later on in this course we will see how to remedy both of these problems using **ctdb**.



References

- Red Hat Storage Server Administration Guide
- Section 9.3: CIFS



Performance Checklist

Configuring CIFS Clients

Lab Overview: In this exercise you will add a Samba-only user to your **cXn1** machine then mount your **distreplvol** volume on your **desktopX** machine using the CIFS protocol.

Success Criteria: You have successfully completed this lab when you have mounted your **distreplvol** on your **desktopX** machine using CIFS.

Before you begin...

Make sure you still have a running volume called **distreplvol**.

- ☐ 1. On your **cXn1** machine start and enable the **smb** service.
- ☐ 2. On your **cXn1** machine create a samba-only user called **cifsuser** with a Samba password of **redhat**. (Samba-only means the user should not be able to log into your machine using any other protocol than CIFS.)
- ☐ 3. Give your new user write permissions on the **distreplvol** volume. One of the ways to do this is to change the permissions on the automatic mount done on **/mnt/samba/distreplvol** on any of your storage nodes.
- ☐ 4. On your **desktopX** system create a mount point called **/mnt/distreplvol**.
- ☐ 5. On your **desktopX** machine *temporarily* mount your **distreplvol** volume on **/mnt/distreplvol** using the CIFS protocol. For authentication you can use the **cifsuser** user you have just created with the password **redhat**.
- ☐ 6. On **desktopX** create 10 new files called **file1** through **file10** in **/mnt/distreplvol**.
- ☐ 7. On **cXn1** through **cXn4** inspect the contents of the bricks that make up your **distreplvol** volume (**/bricks/brick5** through **/bricks/brick8** respectively).

What do you notice about the file distribution?

- ☐ 8. Unmount your **distreplvol** volume from **desktopX**.

Volume Options

Using Red Hat Storage Server you can set volume-specific options. These options range from access control to low-level tuning options for volumes. The basic syntax to set an option is **gluster volume set <VOLUME NAME> <OPTION> <VALUE>**.

To reset an option back to its default you can use the following syntax: **gluster volume reset <VOLUME NAME> <OPTION>**

Run the following command to query all the options that have been set for a volume: **gluster volume info <VOLUME NAME>**

The following command requests an (almost) complete list of all built-in options along with their default value and a short explanation: **gluster volume set help**.

Some useful options and their explanations are listed below:

Option	Default	Description
auth.allow	*	A comma-separated list of clients that are allowed to connect to this volume using the native client or NFSv3. Wildcards can be used, e.g. 10.0.0.* .
auth.reject		A comma-separated list of clients that should never be allowed to connect to this volume using either the native client or NFSv3. If a client is listed in both auth.allow and auth.reject the client will be rejected. Wildcards are allowed.
nfs.disable		When set to on this volume will not be exported over NFSv3.
nfs.volume-access	read-write	This option determines whether the volume is exported read-only or read-write over NFSv3. To export the volume read-only set this option to read-only .

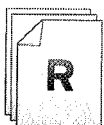
You can also set user-defined options for volumes. These options will start with **user..** Unlike built-in options, user-defined options can *not* be reset.

User-defined options can be used in your own (hook-)scripts. The samba hook-scripts, for example, query the option **user.cifs** and will not export a volume over CIFS if it is set to **disable**.



Important

When changing access rules, you will typically need to stop then start your volume in order for the changes to become active.



References

- Red Hat Storage Server Administration Guide
 - Section 10.1: Tuning Volume Options



Performance Checklist

Setting Volume Options

Lab Overview: In this exercise you will set, and reset, volume options for your **firstvol** volume.

Before you begin...

Make sure you still have a working **replvol** volume that is persistently mounted on **/mnt/replvol** on your **cXn5** machine.

- ☐ 1. *Temporarily* unmount your **replvol** volume from your **cXn5** machine.
- ☐ 2. Set the **nfs.disable** option to **On** for your **replvol** volume.
- ☐ 3. Inspect the volume options for your **replvol** volume.
- ☐ 4. On your **cXn5** machine, attempt to mount **/mnt/replvol** again. This should fail.
- ☐ 5. Reset the **nfs.disable** option for your **replvol** volume back to its default setting.
- ☐ 6. Display the volume options for your **replvol** volume.
- ☐ 7. Attempt to mount **/mnt/replvol** on your **cXn5** machine again. This time it should work.

Unit Test



Case Study

Limit Access to Volumes

Lab Overview: In this exercise you will limit access to your **replvol** volume.

Success Criteria: You will have successfully completed this lab when only clients in the **192.168.0.0/24** subnet can connect to your **replvol** volume, with the exception of your **desktopX** machine.

Before you begin...

Make sure you still have a working **replvol** volume.

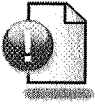
You have been asked to limit access to your **replvol** volume to only clients in the **192.168.0.0/24** subnet.

Note: Reconfiguring access restrictions requires stopping and starting the volume.

How would you address the case study described above? Take notes on your process in the space below and then implement it.



Personal Notes



Unit Summary

Native Client

- Install and Use the Native Client

NFS Clients

- Using NFSv3 Clients

CIFS Clients

- Use CIFS clients

Volume Options

- Configure volume-specific options



UNIT 7

ACLs AND QUOTAS

Introduction

Unit Details	
Unit Goal	Implement POSIX ACLs and quotas
Unit Sections	<ul style="list-style-type: none">• POSIX ACLs• Quotas
Hands-On Activities	<ul style="list-style-type: none">• Using POSIX ACLs• Working with Quotas
Unit Test	Quotas and ACLs

POSIX ACLs

Red Hat Storage volumes support the use of *POSIX ACLs* to allow for fine-grained access control to files and directories.

You can use the mount option **acl** to enable POSIX ACLs when using the native client. When using the NFSv3 client any existing ACLs will be enforced, but you will not be able to query or set them. The CIFS client allows you to query ACLs and they will be enforced, but you will not be able to set them.



Important

At the time of writing, the automatic mount on `/mnt/samba/<VOLUME NAME>` used to export your volumes over CIFS does not use the **acl** mount option. Clients will be able to query any existing ACLs if you set up your mounts and CIFS exports manually using the **acl** mount option.

Working with POSIX ACLs

Two commands work with POSIX ACLs: **getfacl** and **setfacl**.

getfacl FILE... will display any existing ACLs on the files and directories specified.

setfacl -m <ACL> FILES... modifies and sets ACLs. ACLs consist of three parts: **u** | **g**:<user-or-group>:<permissions>, where a leading **u** indicates a User-ACL, and a leading **g** indicates a Group-ACL.

Prepending a **d**: in front of an ACL will turn it into a *default* ACL. Default ACLs can only be set on directories and do not actually set any permissions themselves. Default ACLs are used to set ACLs that should automatically be applied to any new files or directories created in the directory on which the default ACL is set.



References

- Red Hat Storage Server Administration Guide
 - Section 9.5: POSIX Access Control Lists



Performance Checklist

Using POSIX ACLs

Lab Overview: In this exercise you will reconfigure your mount of **firstvol** on your **cXn5** machine to use ACLs.

Success Criteria: You will have successfully completed this lab when you have implemented POSIX ACLs on your **firstvol** volume.

Before you begin...

Make sure you still have your **firstvol** volume mounted on **/mnt/firstvol** on your **cXn5** machine using the native client.

- ☐ 1. Add the **acl** mount option to **/etc/fstab** on **cXn5** for the **firstvol** volume.
- ☐ 2. Re-mount your **firstvol** volume on **cXn5**.
Note: Since the native client currently does not support the **remount** mount option, you will have to unmount then mount your volume.
- ☐ 3. Create a new user on **cXn5** named **acluser**.
- ☐ 4. Create a new directory on **cXn5** named **/mnt/firstvol/supersecret** that is owned by the user **root** and the group **root**, and has permissions set to **0700**.
- ☐ 5. As the **acluser** user on **cXn5**, try to list the contents of the **/mnt/firstvol/supersecret** directory. This should fail.
- ☐ 6. Add an ACL to the **/mnt/firstvol/supersecret** directory allowing **acluser** full access (read, write, and execute).
- ☐ 7. Also add an ACL to **/mnt/firstvol/supersecret** allowing **acluser** full access to any new files and directories created underneath it.
- ☐ 8. Verify the ACLs you have just created with **getfacl**.
- ☐ 9. Attempt to create a new file called **/mnt/firstvol/supersecret/launchcodes.txt** as **acluser** then verify the ACLs on the newly created file.

Quotas

Red Hat Storage Server supports setting quotas on individual directories on a volume. This allows an administrator to control where storage is being used. Unlike the file system quotas you might be used this is done on a directory basis, and not a user/group basis. This is similar to *Project Quotas* on a XFS file system. Use the following command to enable quotas for a volume:

```
[root@server ~]# gluster volume quota <VOLUME> enable
```

To set a quota on a directory use the command below:

```
[root@server ~]# gluster volume quota <VOLUME> limit-usage <path-on-volume> size
```

Note that for **<size>** you can use units like **MB**, **GB**, and **PB**.



Specifying Path Names

When specifying the path name of a directory on a volume, you must only specify the directories relative to the top level directory of the volume it resides on, but the path name must start with a leading slash. For example, if you have a volume mounted on **/mnt/serenity** and you want to set a directory limit on **/mnt/serenity/alliance**, you would specify the path name as **/alliance**.



Note

You can set limits for directories that do not yet exist. The limits will be enforced the moment the directory is created.

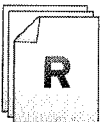
The following command displays the quota usage for a volume:

```
[root@server ~]# gluster volume quota <VOLUME> list
```



Note

Quota information is not updated atomically, therefore it is possible for a directory to go slightly over quota. The **features.quota-timeout** volume option controls the frequency the quota information is synced.



References

- Red Hat Storage Server Administration Guide
 - Chapter 12: Managing Directory Quota



Performance Checklist

Working with Quotas

Lab Overview: In this exercise you will enable, set, and enforce quotas on your **replvol** volume.

Success Criteria: You will have successfully completed this lab when you enabled quotas for your **replvol** volume with a 256M quota set for **replvol/quotadir**.

Before you begin...

Make sure the **replvol** volume is active and mounted on **cXn5**.

- ☐ 1. Enable quotas for your **replvol** volume.
- ☐ 2. Set a 256 MiB limit for the **/quotadir** directory on your **replvol** volume.
- ☐ 3. From your **cXn5** machine, create the **/mnt/replvol/quotadir** directory then create a 128MiB file called **shiny** in that directory.
- ☐ 4. On **cXn1** display a quota overview for your **replvol** volume.
- ☐ 5. On your **cXn5** machine attempt to create a 512 MiB file called **/mnt/replvol/quotadir/toolarge**. This should fail after approximately 128 MiB.

Unit Test



Case Study

Quotas and ACLs

Lab Overview: In this exercise you will create a shared directory with both quotas and ACLs on your **firstvol** volume.

Success Criteria: You will have successfully completed this lab when you have created a shared directory on your **firstvol** volume using the criteria below.

Before you begin...

Make sure you still have a working **firstvol** volume mounted on **/mnt/firstvol** on your **cXn5** machine.

You have been asked to create a shared directory for the (new) **architects** group. This new directory should be created with the following criteria:

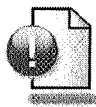
- Location: **firstvol/architects**
- Ownership: **root:architects**
- Permissions:
 - All files and directories in **architects** should be owned by the group **architects**.
 - People in the **architects** group should have full access to all files and directories in **firstvol/architects**, including any newly created files and directories.
 - People in the (new) **managers** group should have read-only access to **firstvol/architects** and any new files and/or directories created there. (This includes *execute* permission for directories.)
 - People outside of the **architects** and **managers** groups should have no access to the **firstvol/architects** directory or any files and directories located below that directory. (It is understood that **root** has full access.)
 - Any new files or directories created under the **firstvol/architects** directory should automatically belong to the **architects** group. You will probably want to use **setgid** permission to fulfill this requirement.
- Disk usage for the **firstvol/architects** directory should be limited to a maximum of 512 MiB.

As part of these criteria you will have to create both the **architects** and **managers** groups on your **cXn5** machine.

How would you address the case study described above? Take notes on your process in the space below and then implement it.



Personal Notes



Unit Summary

POSIX ACLs

- Use POSIX ACLs

Quotas

- Enable directory quotas



UNIT 8

EXTENDING VOLUMES

Introduction

Unit Details	
Unit Goal	Grow storage volumes on-line
Unit Sections	<ul style="list-style-type: none">• Growing Volumes
Hands-On Activities	<ul style="list-style-type: none">• Extending a Volume
Unit Test	Extending a Volume

Growing Volumes

You can extend a running volume, without causing any downtime, by adding bricks to it. When adding bricks you must always add enough bricks so that the current layout can be maintained. For example, when extending a volume with **replica** set to 2, you must add a multiple of 2 bricks at once. Likewise, when extending a 4x4 striped replica you must add at least 16 bricks or a multiple thereof (or four bricks when extending the replica count to 5).

It is also possible to grow the number of replicas when extending a volume. In order to do this specify the **replica <new-number-of-replicas>** option when extending. The amount of bricks added must match the intended layout.



Warning

It is *not* possible to change the **stripe** count when extending a volume. Attempting to do so will result in a warning. Ignoring this warning will definitely lead to data loss and a non-functioning volume.

Use the **gluster volume add-brick <VOLUME> [replica <new-replica-count>] BRICK... command** to extend a volume.



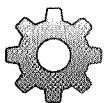
Note

It is also possible to remove bricks from a volume. Refer to the Red Hat Storage Server Administration Guide for the steps to this process.

Rebalancing Volumes

After extending or shrinking a volume it is typically necessary to *rebalance* the volume. Rebalancing a volume will move files between bricks to match the new layout.

To start a rebalance operation use the command **gluster volume rebalance <VOLUME> start**. Once a rebalance has been started you can follow its progress using the **gluster volume rebalance <VOLUME> status** command.



Demonstration

Extending a Volume

In this demonstration your instructor will demonstrate how to extend and rebalance a volume.

1. Make sure we have some files to rebalance.

```
[root@c0n1 ~]# touch /mnt/samba/demovol/file{000..499}
```

2. Add a third host to our trusted storage pool.

```
[root@c0n1 ~]# gluster peer probe c0n3.example.com
Probe successful
```

3. Create a new brick on the new host.

```
[root@c0n3 ~]# lvcreate -L 2G -n brick3 vg_bricks
Logical volume "brick3" created
[root@c0n3 ~]# mkfs.xfs -i size=512 /dev/vg_bricks/brick3
meta-data=/dev/vg_bricks/brick3 isize=512    agcount=4, agsize=131072 blks
       =                               sectsz=512   attr=2
data     =                               bsize=4096   blocks=524288, imaxpct=25
       =                               sunit=0      swidth=0 blks
naming   =version 2                       bsize=4096   ascii-ci=0
log      =internal log                   bsize=4096   blocks=2560, version=2
       =                               sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                           extsz=4096   blocks=0, rtextents=0
[root@c0n3 ~]# mkdir -p /bricks/brick3
[root@c0n3 ~]# echo "/dev/vg_bricks/brick3 /bricks/brick3 xfs defaults 0 2" >> /etc/
fstab
[root@c0n3 ~]# mount -a
```

4. Add the new brick to your volume.

```
[root@c0n1 ~]# gluster volume add-brick demovol c0n3.example.com:/bricks/brick3
Add Brick successful
```

5. Start a rebalance operation.

```
[root@c0n1 ~]# gluster volume rebalance demovol start
Starting rebalance on volume demovol has been successful
```

6. Monitor the progress of the rebalance operation.

```
[root@c0n1 ~]# gluster volume rebalance demovol status
```

Node	Rebalanced-files	size	scanned	failures	status
localhost	79	0	594	0	completed
c0n2.example.com	79	0	657	0	completed
c0n3.example.com	0	0	580	0	completed



References

Red Hat Storage Server Administration Guide

- Section 10.2: Expanding Volumes
- Section 10.5: Rebalancing Volumes



Performance Checklist

Extending a Volume

Lab Overview: In this exercise you will extend your **firstvol** volume to use two extra bricks.

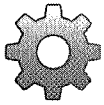
Success Criteria: You will have successfully completed this lab when your **firstvol** has a total of four bricks.

Before you begin...

Make sure you still have a working **firstvol** volume.

- ☐ 1. Begin by creating two 2 GiB bricks, **/dev/vg_bricks/brick9** on **cXn1**, mounted on **/bricks/brick9**, and **/dev/vg_bricks/brick10** on **cXn2**, mounted on **/bricks/brick10**.
- ☐ 2. Add your two new bricks to your **firstvol** volume. Make sure to keep the volume type as *Distributed*.
- ☐ 3. Initiate a *rebalance* of your **firstvol** volume.
- ☐ 4. Wait for the rebalance to complete.
- ☐ 5. Inspect the contents of your different bricks, what do you notice?

Unit Test



Case Study

Extending a Volume

Lab Overview: In this exercise you will grow your **replvol1** volume into a 2x2 Distributed-Replicate volume.

Success Criteria: You will have successfully completed this lab when your **replvol1** is a 2x2 Distributed-Replicate volume.

Before you begin...

Make sure you still have a working **replvol1** volume.

You have been asked to grow your **replvol1** volume into a 2x2 Distributed-Replicate volume.

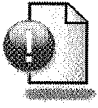
In order to accomplish this task you should use two new 2 GiB bricks. On **cXn3** you should add a new brick on **/dev/vg_bricks/brick11**, mounted on **/brick/brick11**, and on **cXn4** you should add a new brick on **/dev/vg_brick/brick12**, mounted on **/bricks/brick12**.

Any new files written to your brick should be distributed according to the new layout, and any existing files should be re-distributed.

How would you address the case study described above? Take notes on your process in the space below and then implement it.



Personal Notes



Unit Summary

Growing Volumes

- Extend a Volume



UNIT 9

IP FAILOVER

Introduction

Unit Details	
Unit Goal	Configure IP failover
Unit Sections	<ul style="list-style-type: none">• IP Failover with CTDB
Hands-On Activities	None
Unit Test	Configuring IP Failover

IP Failover with CTDB

In order to facilitate automatic IP failover when using NFSv3 or CIFS, you can use *CTDB*, a Clustered implementation of the **Samba** Trivial DataBase.

After configuring a *replicated* volume for the storage of our clustered database and configuration files you can configure the *ctdb* service as well as the hook scripts necessary to automatically configure CIFS exports to use the clustered database for user authentication.



Important

Since Amazon EC2 does not support the use of floating IP addresses you cannot use CTDB on Amazon EC2.



Demonstration

Configuring CTDB

In this demonstration your instructor will demonstrate the steps necessary to configure **CTDB** for automatic IP failover.

1. First you will need to configure a replicated volume to store your databases and configuration files.
 1. Create the bricks necessary to form your new volume. It is recommended to replicate your volume across at least three bricks on different machines.

```
[root@c0n1 ~]# lvcreate -L 256M -n ctdbbrick1 vg_bricks
Logical volume "ctdbbrick1" created
[root@c0n1 ~]# mkfs.xfs -i size=512 /dev/vg_bricks/ctdbbrick1
meta-data=/dev/vg_bricks/ctdbbrick1 isize=512    agcount=4, agsize=16384 blks
       =                               sectsz=512   attr=2
data     =                               bsize=4096   blocks=65536, imaxpct=25
       =                               sunit=0      swidth=0 blks
naming   =version 2                     bsize=4096   ascii-ci=0
log      =internal log                 bsize=4096   blocks=1200, version=2
       =                               sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                         extsz=4096   blocks=0, rtextents=0
[root@c0n1 ~]# mkdir /bricks/ctdbbrick1
[root@c0n1 ~]# echo "/dev/vg_bricks/ctdbbrick1 /bricks/ctdbbrick1 xfs defaults 0
2" >> /etc/fstab
[root@c0n1 ~]# mount -a
```

Repeat this procedure for all bricks.

2. Combine your new bricks into a replicated volume. Do *NOT* start this volume yet.

```
[root@c0n1 ~]# gluster volume create ctdbmeta replica 3 \
> c0n1.example.com:/bricks/ctdbbrick1 \
> c0n2.example.com:/bricks/ctdbbrick2 \
> c0n3.example.com:/bricks/ctdbbrick3
Creation of volume ctdbmeta has been successful. Please start the volume to
access data.
```

- On all of your nodes update the `/var/lib/glusterd/hooks/1/start/post/S29CTDBsetup.sh` and `/var/lib/glusterd/hooks/1/stop/pre/S29CTDB-teardown.sh` scripts to set the `META=` variable to the name of your newly created volume.

```
[root@c0nY ~]# sed -i 's/^META="all"/META="ctdbmeta"/' /var/lib/glusterd/hooks/1/
start/post/S29CTDBsetup.sh
[root@c0nY ~]# sed -i 's/^META="all"/META="ctdbmeta"/' /var/lib/glusterd/hooks/1/
stop/pre/S29CTDB-teardown.sh
```

- Start your new volume.

```
[root@c0n1 ~]# gluster volume start ctdbmeta
Starting volume ctdbmeta has been successful
```

This should have automatically mounted your new volume on `/gluster/lock` on all your nodes.

- Create the `/gluster/lock/ctdb` configuration file with the following contents. For a complete overview of possible options you can check the existing file at `/etc/sysconfig/ctdb` on any of your nodes.

```
CTDB_RECOVERY_LOCK=/gluster/lock/lockfile
CTDB_PUBLIC_ADDRESSES=/etc/ctdb/public_addresses
CTDB_MANAGES_SAMBA=yes
CTDB_NODES=/etc/ctdb/nodes
```

- On all of your nodes create a symlink from `/etc/sysconfig/ctdb` to `/gluster/lock/ctdb`. You will probably have to remove the existing `/etc/sysconfig/ctdb` file first.

```
[root@c0nY ~]# mv /etc/sysconfig/ctdb{,.bak}
[root@c0nY ~]# ln -s /gluster/lock/ctdb /etc/sysconfig/ctdb
```

- Create a new file called `/gluster/lock/nodes` and list the IP addresses of your nodes in that file, one IP address per line.

```
[root@c0n1 ~]# for IP in 192.168.0.{1..3}; do echo ${IP} >> /gluster/lock/nodes; done
```

- On all of your nodes create a symlink from `/etc/ctdb/nodes` to `/gluster/lock/nodes`.

```
[root@c0nY ~]# ln -s /gluster/lock/nodes /etc/ctdb/nodes
```

- Create a new `/gluster/lock/public_addresses` file listing the *floating* IP addresses you wish to use on your cluster, their prefix, and the network interface that should be used for that IP address.

```
[root@c0n1 ~]# echo "192.168.0.6/24 eth0" > /gluster/lock/public_addresses
```

Each floating IP address will be in use on only one machine at a time, but when that machine fails another machine in your cluster will take over that floating IP address maintaining service on that IP address for your clients.

9. On all of your nodes create a symlink from `/etc/ctdb/public_addresses` to `/gluster/lock/public_addresses`.

```
[root@c0nY ~]# ln -s /gluster/lock/public_addresses /etc/ctdb/public_addresses
```

10. Make sure the **smb** service is disabled on all of your nodes. CTDB will take care of starting it for us in our configuration.

```
[root@c0nY ~]# chkconfig smb off
```

11. Start the **ctdb** service on all of your nodes.

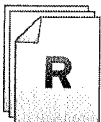
```
[root@c0nY ~]# service ctdb start
```

12. Verify your CTDB configuration. Some useful commands include **ctdb ip -n all**, **ctdb status**, **ctdb ping -n all**, and **ctdb listnodes**.



Note

When using CTDB with Samba, the Samba user account database will be moved to `/var/ctdb/persistent/*tdb`. You will need to recreate any Samba users you had before since the existing database has not been migrated automatically, but you will only need to do so from one machine. However you must make sure all nodes have local user accounts for your Samba users, e.g. by using a central LDAP server.



References

How to configure Red Hat Storage with IP-address failover using CTDB?
<https://access.redhat.com/knowledge/solutions/66020>

Red Hat Storage Server Administration Guide

- Section 9.4: Configure Automated IP Failover for NFS and CIFS

Unit Test



Performance Checklist

Configuring IP Failover

Lab Overview: In this exercise you will configure IP failover for your NFS and CIFS exports.

Success Criteria: You will have successfully completed this exercise when **CTDB** is fully configured and running.

Before you begin...

Make sure all of your nodes are still operational.

- ☐ 1. First we will need a replicated volume to store our metadata for **CTDB**. On your **cXn1** and **cXn2** machines create a new 256 MiB logical volume called **brickY+12**, where **Y+12** is the node number plus 12, format them with an XFS file system, and mount them on **/bricks/brick13** and **/bricks/brick14** respectively.
- ☐ 2. Combine your two new bricks into a two-way replicated volume called **ctdbmeta**. Do *not* start this volume right now.
- ☐ 3. Verify your new volume using **gluster volume info**.
- ☐ 4. On all four of your nodes, (**cXn1** through **cXn4**), change **META="all"** to **META="ctdbmeta"** in both **/var/lib/glusterd/hooks/1/start/post/S29CTDBsetup.sh** and **/var/lib/glusterd/hooks/1/stop/pre/S29CTDB-teardown.sh**.
- ☐ 5. Start your **ctdbmeta** volume.
- ☐ 6. Your new volume should now be mounted on **/gluster/lock**. Verify that this is the case on all four of your nodes.
- ☐ 7. Create the **/gluster/lock/ctdb** file with the following contents:

```
CTDB_RECOVERY_LOCK=/gluster/lock/lockfile
CTDB_PUBLIC_ADDRESSES=/etc/ctdb/public_addresses
CTDB_MANAGES_SAMBA=yes
CTDB_NODES=/etc/ctdb/nodes
```

- ☐ 8. Create a new **/gluster/lock/nodes** file and list the IP addresses of your four nodes in it, one per line.
- ☐ 9. Create a new file called **/gluster/lock/public_addresses** with the following content:

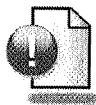
```
192.168.0.X6/24 eth0
```

This means that we want to use the floating IP address **192.168.0.X6**, with a prefix (subnetmask) of **24** and it should be assigned to the **eth0** interface when in use.

- ❑ 10. On all four your nodes create a symlink from `/etc/sysconfig/ctdb` to `/gluster/lock/ctdb`, from `/etc/ctdb/nodes` to `/gluster/lock/nodes`, and from `/etc/ctdb/public_addresses` to `/gluster/lock/public_addresses`. You might have to remove an existing `/etc/sysconfig/ctdb` file first.
- ❑ 11. Start the **ctdb** service on all four of your nodes.
- ❑ 12. Now you can test if your failover works. On your **cXn5** machine, unmount `/mnt/replvol` then change the `/etc/fstab` entry for **replvol** to use the floating IP address you configured (**192.168.0.X6**).
- ❑ 13. Mount the **replvol** volume on **cXn5**, then verify that you can access the volume.
- ❑ 14. Determine which of your hosts currently has the floating IP address. To do this use the **ip a s eth0** command on all four of your nodes to determine which host has the floating IP address. Another options is to use the **ctdb ip** command.
- ❑ 15. Shutdown the node that currently has the public IP (in our example this is the **cXn1** machine, but you might have to shutdown a different machine).
- ❑ 16. Attempt to access `/mnt/replvol` from your **cXn5** machine. This operation should stall for a number of seconds, then continue.
- ❑ 17. **Important Cleanup:** Restart the node that you shut down during this lab.



Personal Notes



Unit Summary

IP Failover with CTDB

- Configure CTDB for automatic IP failover



UNIT 10

GEO-REPLICATION

Introduction

Unit Details	
Unit Goal	Configure Geo-Replication for disaster recovery
Unit Sections	<ul style="list-style-type: none">• Configuring Geo-Replication
Hands-On Activities	<ul style="list-style-type: none">• Configure Geo-Replication
Unit Test	None

Configuring Geo-Replication

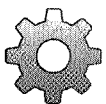
Apart from replicated volumes, which offer *synchronous* replication, Red Hat Storage Server also offers *asynchronous* replication in the form of *Geo-Replication*. Geo-Replication can be used to keep a copy of your data off-site, to protect against data-loss when an entire data center goes down, or to provide fail-over. Geo-Replication can be used over a LAN, a WAN, and over the Internet.

Geo-Replication uses a *Master-Slave* model, and as such data will be replicated one-way, i.e. from the master to the slave. A slave can be a local directory, a local Red Hat Storage volume, or a remote Red Hat Storage volume. In all cases the transport can be secured by tunneling it over an SSH connection.



Warning

Geo-Replication is *not* instantaneous, instead files are checked for changes at regular intervals and synced when a difference is detected.



Demonstration

Configuring Geo-Replication

In this demonstration your instructor will demonstrate how to configure Geo-Replication to a remote Red Hat Storage cluster using an SSH transport and the *mount-broker*.

1. On the master create a new SSH key-pair without a passphrase in `/var/lib/glusterd/geo-replication/secret.pem`.

```
[root@c0n1 ~]# ssh-keygen -P '' -f /var/lib/glusterd/geo-replication/secret.pem
Generating public/private rsa key pair.
Your identification has been saved in /var/lib/glusterd/geo-replication/secret.pem.
Your public key has been saved in /var/lib/glusterd/geo-replication/secret.pem.pub.
The key fingerprint is:
be:c4:03:fd:96:40:d9:f5:05:d2:f5:62:42:f5:e2:c1 root@c0n1.example.com
The key's randomart image is:
+--[ RSA 2048 ]-----+
|           +oooo|
|        o o +.o.|
|       o . . E o|
|      o   + + |
|     . S   . |
|    + o .   |
|   = +     |
|  . +      |
|  .        |
+-----+
```

2. Modify the public part of your new key-pair so it can only be used to execute the `/usr/libexec/glusterfs/gsyncd` command. This can be done by inserting the following at the start of the `/var/lib/glusterd/geo-replication/secret.pem.pub` file:

```
command="/usr/libexec/glusterfs/gsyncd"
```

```
[root@c0n1 ~]# sed -i 's|^|command="/usr/libexec/glusterfs/gsyncd" |' /var/lib/
glusterd/geo-replication/secret.pem.pub
```

3. On your target (slave) machine add a user and a group that will be used for the SSH sync connection. Since we are going to use an unprivileged user, later we will configure the *mount-broker* to allow that user full access to the slave volume.

```
[root@c0n4 ~]# groupadd geogroup
[root@c0n4 ~]# useradd -G geogroup geouser
[root@c0n4 ~]# echo redhat | passwd --stdin geouser
```

4. Copy the public key you created earlier to the account you created on the slave.

```
[root@c0n1 ~]# ssh-copy-id -i /var/lib/glusterd/geo-replication/secret.pem.pub
geouser@c0n4.example.com
```

5. On your slave create a volume that is large enough to hold the data from your master volume, this typically means creating it at the same size.

```
[root@c0n4 ~]# lvcreate -L 66 -n geobrick1 vg_bricks
Logical volume "geobrick1" created
[root@c0n4 ~]# mkfs.xfs -i size=512 /dev/vg_bricks/geobrick1
meta-data=/dev/vg_bricks/geobrick1 isize=512    agcount=4, agsize=393216 blks
       =                               sectsz=512   attr=2
data     =                               bsize=4096   blocks=1572864, imaxpct=25
       =                               sunit=0      swidth=0 blks
naming   =version 2                       bsize=4096   ascii-ci=0
log      =internal log                     bsize=4096   blocks=2560, version=2
       =                               sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                             extsz=4096   blocks=0, rtextents=0
[root@c0n4 ~]# mkdir -p /bricks/geobrick1
[root@c0n4 ~]# echo "/dev/vg_bricks/geobrick1 /bricks/geobrick1 xfs defaults 0 2"
>> /etc/fstab
[root@c0n4 ~]# mount -a
[root@c0n4 ~]# gluster volume create geoslave c0n4.example.com:/bricks/geobrick1
Creation of volume geoslave has been successful. Please start the volume to access
data.
[root@c0n4 ~]# gluster volume start geoslave
Starting volume geoslave has been successful
```

6. Now you will need to configure the *mount-broker*. The mount-broker allows specified unprivileged users to use Red Hat Storage volumes as if they were a privileged user.

First create the directory to act as a root for the mount-broker.

```
[root@c0n4 ~]# mkdir /var/mountbroker-root
```

7. Tell the **glusterd** daemon on your slave to use the mount-broker and which users and groups are allowed to use it.

Add the following lines to the **volume management** section of **/etc/glusterfs/glusterd.vol**, then restart the **glusterd** service.

```
option mountbroker-root /var/mountbroker-root
option mountbroker-geo-replication.geouser geoslave
option geo-replication-log-group geogroup
```

The **option.mountbroker-geo-replication.geouser geoslave** line gives the **geouser** user mount-broker access to the **geoslave** volume.

```
[root@c0n4 ~]# service glusterd restart
```

8. Now you are ready to initiate your geo-replication using the mount-broker and SSH.

```
[root@c0n1 ~]# gluster volume geo-replication demovol
geouser@c0n4.example.com::geoslave start
Starting geo-replication session between demovol & geouser@c0n4.example.com::geoslave
has been successful
```

9. Verify the status of replication:

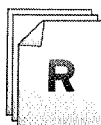
```
[root@c0n1 ~]# gluster volume geo-replication demovol
geouser@c0n4.example.com::geoslave status
```

MASTER	SLAVE	STATUS
demovol	geouser@c0n4.example.com::geoslave	starting...

10. Wait for the initial synchronization to complete:

```
[root@c0n1 ~]# gluster volume geo-replication demovol
geouser@c0n4.example.com::geoslave status
```

MASTER	SLAVE	STATUS
demovol	geouser@c0n4.example.com::geoslave	OK



References

- Red Hat Storage Server Administration Guide
- Chapter 11: Managing Geo-Replication



Performance Checklist

Configure Geo-Replication

Lab Overview: In this exercise you will configure Geo-Replication between your **firstvol** volume on **cXn1** and a slave volume, **slavevol**, on **cXn5**

Success Criteria: You will have successfully completed this lab when geo-replication is working for your **firstvol** volume.

Before you begin...

Make sure your **firstvol** volume is still working.

- ☐ 1. Create an SSH key that will be used for geo-replication. Be sure you store it in **/var/lib/glusterd/geo-replication/secret.pem** on **cXn1**.
- ☐ 2. Edit **/var/lib/glusterd/geo-replication/secret.pem.pub** so that only the **/usr/libexec/glusterfs/gsync** command can be executed when using this key.
- ☐ 3. Create a new group on **cXn5** called **geogroup** and a new user called **geoaccount** that is a member of the group **geogroup**. Set the password for **geoaccount** to **redhat**.
- ☐ 4. From your **cXn1** machine, copy the SSH publickey you just created to the **geoaccount** account on **cXn5**.
- ☐ 5. On **cXn5** create a new 8 GiB logical volume called **slavebrick1** in the **vg_bricks** volume group, format it with an XFS file system with 512 byte inodes, and persistently mount it on **/bricks/slavebrick1**
- ☐ 6. On **cXn5** create a new volume called **slavevol** using the brick you just created, then start it.
- ☐ 7. On **cXn5** create a new directory, **/var/mountbroker-root**, to act as a root mount point for the mount broker.
- ☐ 8. On **cXn5** add the following lines to the **volume management** section of **/etc/glusterfs/glusterd.vol**:

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

The lines allow **geoaccount** access to the **slavevol** volume using the mount broker and allow members of the group **geogroup** to read the log files.

- ☐ 9. Restart the **glusterd** daemon on **cXn5** to activate your changes.
- ☐ 10. From **cXn1** initiate geo-replication between the **firstvol** volume and the **slavevol** volume on **cXn5** using the **geoaccount** account and the mount broker.
- ☐ 11. Inspect the status of your replication.
- ☐ 12. Wait for the initial replication to finish.

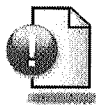


Note

Due to a bug it can happen that your Geo-Replication reports a state of **faulty**, even when you have configured everything as it should be. If this happen to you, check if all your files are synchronized, including any new files you might create. If files do get replicated, but the status stays at **faulty** just pretend that the status reads **OK** for this lab.



Personal Notes



Unit Summary

Configuring Geo-Replication

- Prepare for Disaster Recovery using Geo-Replication



UNIT 11

UNIFIED FILE AND OBJECT STORAGE

Introduction

Unit Details	
Unit Goal	Configure Swift Object access
Unit Sections	<ul style="list-style-type: none">• Introduction to Swift• Configuring Swift• Testing Swift
Hands-On Activities	<ul style="list-style-type: none">• Configuring Swift• Testing Swift
Unit Test	None

Introduction to Swift

Red Hat Storage Server contains an implementation of the OpenStack Object Storage interface using REST (Representational State Transfer). This allows developers to use a standardized API to talk to Red Hat Storage Servers for Object Storage and Retrieval. While you use this protocol the same volume will still remain accessible as a normal POSIX compliant file system to other clients.

This technology allows developers to create applications that use storage using a standardized API. Possible applications include file-sharing services and NoSQL applications. Almost every single layer of the stack can be customized using *WSGI* applications, one could for example add a virus scanner that automatically scans all files being uploaded and downloaded, or add a custom authentication/authorization system.

Terminology

Proxy Server

The *Proxy Server* is the piece of software that interfaces between the REST API and the underlying Red Hat Storage volume.

Objects

An object is the basic storage entity, combined with any optional metadata describing the object. When an object is uploaded (or downloaded) it is transferred as-is, without any compression or encryption. Objects are stored as files on Red Hat Storage volumes.

Containers

A container is a storage compartment for your data. Red Hat Storage Unified File and Object Storage maps containers to directories on a Red Hat Storage Volume.

Accounts and User

Since OpenStack Object Storage is designed to be used by many different storage consumers at a time, there is a distinction between *accounts* and *users*. An account is mapped directly to a Red Hat Storage volume, while a user has authentication credentials, and can access one account. One account can have multiple users associated with it.

Authentication and Authorization

Users must authenticate against an *Authentication Service*. Doing so will give them an *Authentication Token* which they can use for further access. There are many authentication services that you can use. A basic one that is shipped with Red Hat Storage Server is called **tempauth**.

By default each admin user associated with an account has full access to all objects and containers in that account. Non-admin users can be granted read and/or write privileges on objects and containers using Access Control Lists (ACLs).

Advantages of Unified File and Object Storage

Using the Red Hat Storage Server implementation of the OpenStack Object Storage API has a couple of advantages over using the OpenStack Swift implementation:

- No limit on file sizes (OpenStack Swift limits the object size to 5 GiB)
- Unified view of your objects as both an object store and a POSIX compliant file system

- High Availability
- Scalability
- Replication
- Elastic Volume Management



References

Red Hat Storage Server Administration Guide

- Section 18.1: Components of Object Storage
- Section 18.2: Advantages of using Unified File and Object Storage

Configuring Swift

The main configuration for Unified File and Object Storage is done in the `/etc/swift/proxy-server.conf` file. After editing this file you will need to copy it to all the hosts in your cluster that you wish to use for Swift.

Adding Users

When you are using the **tempauth** authentication service, each user you wish to tie to an account must be defined in the `[filter:tempauth]` section of `/etc/swift/proxy-server.conf`. The syntax for file is (one user per line):

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

Remember that **<account>** should be the name of the volume that you wish to create the user for.

If you add the optional keyword **.admin** at the end of a user line, that user will be considered an administrator for that account. An administrator can set access controls for objects and containers.

memcached

To allow every machine in your cluster to authenticate users and have that authentication honored by the other machines in your cluster, the authentication cookies provided to your users will be stored in a distributed in-memory cache across your machines using the **memcached** service. This means you will have to enable and start the **memcached** service on all your nodes and inform your proxy server of these **memcached** servers.

memcached is a general-purpose distributed memory caching system. It is used by many websites to speed up dynamic web pages by storing data and database objects in RAM.

To inform your proxy server about your **memcached** servers, add the following line to the `[filter:cache]` section of your `/etc/swift/proxy-server.conf` file:

```
memcache_servers = server_IP:11211, ...
```

This lists the IP addresses of all your **memcached** servers and their port numbers (default **11211**), separated by commas.

Starting Swift Services

After copying your completed configuration file to all of your nodes, you will need to start and enable the relevant services on all your nodes. These services are:

- **memcached**
- **gluster-swift-account**
- **gluster-swift-container**
- **gluster-swift-object**
- **gluster-swift-proxy**

A quick way to start all four of the **gluster-swift-*** services (but not enable them) is to run the **swift-init main start** command. This command also allows you to quickly **stop**, **restart**, and **reload** all of the services, as well as to check their **status**.



Important

Remember that you should use inode sizes of at least 1024 bytes for any brick that will be part of a volume being accessed using Unified File and Object Storage.



References

Red Hat Storage Server Administration Guide

- Section 18.4: Configuring Unified File and Object Storage



Performance Checklist

Configuring Swift

Lab Overview: In this exercise you will configure Swift for unified file and object storage.

Success Criteria: You will have successfully completed this lab when you have a working Swift cluster.

Before you begin...

Make sure you still have a working cluster with a working **firstvol** volume.

- ☐ 1. On your **cXn1** machine add an admin user for your **firstvol** volume to the file **/etc/swift/proxy-server.conf** called **ufouser**, with a password of **redhat**.
- ☐ 2. On your **cXn1** machine configure **/etc/swift/proxy-server.conf** to use **memcached** servers on all four of your nodes. Use the default port for **memcached**.
- ☐ 3. From your **cXn1** machine, copy the **/etc/swift/proxy-server.conf** file to your other three nodes.
- ☐ 4. On all four of your nodes enable and start the **memcached** service.
- ☐ 5. On all four of your nodes enable the **gluster-swift-account**, **gluster-swift-container**, **gluster-swift-object**, and **gluster-swift-proxy** services.
- ☐ 6. On all four of your nodes start all of the **gluster-swift-*** services.

Testing Swift

An easy tool to check your Swift configuration is **curl**. The **curl** command allows you to use HTTP **GET**, **PUT**, and **POST** commands from the command-line, using arbitrary headers, and (optionally) showing the full response from the server.

Obtaining an Authentication Token

You must first obtain an *Authentication Token* to perform any actions using Swift. An Authentication token can be obtained by sending a HTTP **GET** request to the **/auth/v1.0** URL on one of your servers, passing in two headers:

```
X-Storage-User: <account>:<user>
X-Storage-Pass: <password>
```

In your response will be a header (use the **-v** option to see those) called **X-Auth-Token**. This contains your authentication token and can be passed in any subsequent request to authenticate.

```
[root@c0n4 ~]# curl -v -H 'X-Storage-User: demovol:demouser' -H 'X-Storage-Pass: redhat'
http://c0n1.example.com:8080/auth/v1.0
* About to connect() to c0n1.example.com port 8080 (#0)
* Trying 192.168.0.1... connected
* Connected to c0n1.example.com (192.168.0.1) port 8080 (#0)
> GET /auth/v1.0 HTTP/1.1
> User-Agent: curl/7.19.7 (x86_64-redhat-linux-gnu) libcurl/7.19.7 NSS/3.13.1.0
zlib/1.2.3 libidn/1.18 libssh2/1.2.2
> Host: c0n1.example.com:8080
> Accept: */*
> X-Storage-User: demovol:demouser
> X-Storage-Pass: redhat
>
< HTTP/1.1 200 OK
< X-Storage-Url: http://127.0.0.1:8080/v1/AUTH_demovol
< X-Storage-Token: AUTH_tk70561a653f784f77b7223c97de8a7504
< X-Auth-Token: AUTH_tk70561a653f784f77b7223c97de8a7504
< Content-Length: 0
< Date: Sun, 25 Nov 2012 14:13:40 GMT
<
* Connection #0 to host c0n1.example.com left intact
* Closing connection #0
```

If you plan to use the command-line to access your storage, it might be useful to store the **X-Auth-Token** header in a variable to keep from copy-pasting it constantly:

```
[root@c0n4 ~]# MYAUTH="X-Auth-Token: AUTH_tk70561a653f784f77b7223c97de8a7504"
```



Important

Remember that in a default configuration the Proxy Server will bind to TCP port **8080**.

Working with Objects and Containers

You can also use an HTTP **GET** command to retrieve a list of containers and objects on a volume. The URL you should request is **/v1/AUTH_<volume>/[<path>]**.

```
[root@c0n4 ~]# curl -v -H "${MYAUTH}" http://c0n1.example.com:8080/v1/AUTH_demo1
* About to connect() to c0n1.example.com port 8080 (#0)
* Trying 192.168.0.1... connected
* Connected to c0n1.example.com (192.168.0.1) port 8080 (#0)
> GET /v1/AUTH_demo1/ HTTP/1.1
> User-Agent: curl/7.19.7 (x86_64-redhat-linux-gnu) libcurl/7.19.7 NSS/3.13.1.0
  zlib/1.2.3 libidn/1.18 libssh2/1.2.2
> Host: c0n1.example.com:8080
> Accept: */*
> X-Auth-Token: AUTH_tk70561a653f784f77b7223c97de8a7504
>
< HTTP/1.1 200 OK
< X-Account-Container-Count: 1
< X-Account-Object-Count: 0
< X-Bytes-Used: 0
< X-Object-Count: 0
< X-Account-Bytes-Used: 0
< X-Type: Account
< X-Container-Count: 1
< Accept-Ranges: bytes
< Content-Length: 14
< Content-Type: text/plain; charset=utf-8
< Date: Sun, 25 Nov 2012 14:25:50 GMT
<
democontainer
* Connection #0 to host c0n1.example.com left intact
* Closing connection #0
```



Important

Swift will not show you any files that live directly under the root of your volume since those are not considered to be part of a container.

You can use the same command to display the contents of a file (object). If you want to download the file you can use the **-o <filename>** or **-O** options.

```
[root@c0n4 ~]# curl -H "${MYAUTH}" http://c0n1.example.com:8080/v1/AUTH_demo1/
democontainer/demoobject
This is a demo object
```

Files can be uploaded with the HTTP **PUT** command. To force **curl** to upload a file without doing any conversion you can use the **--data-binary @<local-file-name>** option.

```
[root@c0n4 ~]# curl -v -X PUT -H "${MYAUTH}" http://c0n1.example.com:8080/v1/
AUTH_demo1/democontainer/newobject --data-binary @/etc/hosts
* About to connect() to c0n1.example.com port 8080 (#0)
* Trying 192.168.0.1... connected
* Connected to c0n1.example.com (192.168.0.1) port 8080 (#0)
> PUT /v1/AUTH_demo1/democontainer/newobject HTTP/1.1
> User-Agent: curl/7.19.7 (x86_64-redhat-linux-gnu) libcurl/7.19.7 NSS/3.13.1.0
  zlib/1.2.3 libidn/1.18 libssh2/1.2.2
> Host: c0n1.example.com:8080
> Accept: */*
> X-Auth-Token: AUTH_tk70561a653f784f77b7223c97de8a7504
> Content-Length: 158
> Content-Type: application/x-www-form-urlencoded
>
< HTTP/1.1 201 Created
```

```
< Content-Length: 118
< Content-Type: text/html; charset=UTF-8
< Etag: 54fb6627dbaa37721048e4549db3224d
< Last-Modified: Sun, 25 Nov 2012 14:46:16 GMT
< Date: Sun, 25 Nov 2012 14:46:17 GMT
<
<html>
  <head>
    <title>201 Created</title>
  </head>
  <body>
    <h1>201 Created</h1>
    <br /><br />
  </body>
* Connection #0 to host c0n1.example.com left intact
* Closing connection #0
```

Working With ACLs

By default admin users have full read and write access to all containers (and the objects stored in those containers). To grant non-admin users read and/or write privileges on a container, use an HTTP **POST** action on the container, including **X-Container-Read** and/or **X-Container-Write** headers. The content of those headers is a comma separated list of users that should be granted read or write privileges.

When requesting information on a container with access rules set, those same headers will be reported back to you.

```
[root@c0n4 ~]# curl -v -X POST -H "${MYAUTH}" -X "X-Container-Read: alice,bob,mary" -H
"X-Container-Write: alice" http://c0n1.example.com:8080/v1/AUTH_demo1/democontainer
* About to connect() to c0n1.example.com port 8080 (#0)
* Trying 192.168.0.1... connected
* Connected to c0n1.example.com (192.168.0.1) port 8080 (#0)
> POST /v1/AUTH_demo1/democontainer HTTP/1.1
> User-Agent: curl/7.19.7 (x86_64-redhat-linux-gnu) libcurl/7.19.7 NSS/3.13.1.0
zlib/1.2.3 libidn/1.18 libssh2/1.2.2
> Host: c0n1.example.com:8080
> Accept: */*
> X-Auth-Token: AUTH_tk70561a653f784f77b7223c97de8a7504
> X-Container-Read: alice,bob,mary
> X-Container-Write: alice
>
< HTTP/1.1 204 No Content
< Content-Length: 0
< Content-Type: text/html; charset=UTF-8
< Date: Sun, 25 Nov 2012 14:54:22 GMT
<
* Connection #0 to host c0n1.example.com left intact
* Closing connection #0
```

You can also set access restriction based on the referring host using the following syntax:
.r:<referrer>. Wildcards are allowed in the referrer name.



References

Red Hat Storage Server Administration Guide

- Section 18.5: Working with Unified File and Object Storage



Performance Checklist

Testing Swift

Lab Overview: In this exercise you will use the **curl** command to test the Swift REST-API.

Success Criteria: You will have successfully completed this lab when you have successfully retrieved and stored files from your **firstvol** volume using the Swift REST API.

Before you begin...

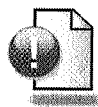
Make sure you have successfully configured Swift and your **firstvol** volume is still operational.

Perform all of the following tasks on your **cXn5** machine.

- ☐ 1. Use **curl** to obtain an authentication token for your **ufouser** user on your **firstvol** volume. Store the entire **X-Auth-Token** header in a shell variable called **MYAUTH**.
- ☐ 2. Use **curl** with the authentication token you just retrieved to request a list of containers on your **firstvol** volume. Remember you should refer to that volume as **AUTH_firstvol** when using Swift.
- ☐ 3. Use **curl** to request the contents of the **supersecret/launchcodes.txt** file on your **firstvol** volume.
- ☐ 4. Use **curl** to create a new file on your **firstvol** volume called **/supersecret/passwd**. Upload the contents of your **/etc/passwd** file on **cXn5** into that new file.
- ☐ 5. Grant the *imaginary* users **alice**, **bob**, and **mary** write access to the **/supersecret** directory on **firstvol**.



Personal Notes



Unit Summary

Introduction to Swift

- Explain Swift
- Compare Swift with OpenStack

Configuring Swift

- Configure Swift

Testing Swift

- Use **curl** to access Swift



UNIT 12

TROUBLESHOOTING

Introduction

Unit Details	
Unit Goal	Perform basic troubleshooting tasks
Unit Sections	<ul style="list-style-type: none">• Troubleshooting
Hands-On Activities	<ul style="list-style-type: none">• Investigate Self-Heal
Unit Test	Replacing a Brick

Troubleshooting

Log Files

Red Hat Storage Server maintains a large amount of log files. These log files are all stored in `/var/log/glusterfs`. This directory contains log files for your native client mounts (also on clients), individual log files for all your bricks (in the **bricks** sub-directory), volume log files, as well as log files for Geo-Replication, rebalance operations, and more.



Note

Log files are rotated automatically on a weekly basis. To force the rotation of the logs for a specific volume or brick you can use the **gluster volume log rotate <VOLUME> [<BRICK>]** command.

Pro-Active Self Heal

When a brick in a replicated volume has been down it will need to be brought back in sync with the other replicas in the volume. Red Hat Storage Server does this by running a *Self-Heal* daemon on each of your nodes. Every ten minutes the self-heal daemon will check if any files in your replicated volumes need healing, and if so replicate them from another brick.

Run the **gluster volume heal <VOLUME> info [healed|heal-failed|split-brain]** command to view the status of the self-heal process for a specific volume. Using one of the three optional arguments will give more detailed information on files that have already been healed, or for which a heal was not possible due to a conflict.

If you do not want to wait for self-heal process to start you can also trigger one manually using the **gluster volume heal <VOLUME> [full]** command. All files will be synchronized when you specify the **full** option, not just those that got created, changed, or removed, since the brick went down.

Replacing Bricks

In some situations you might want to replace a (faulty) brick for another one. This can be useful in cases where the original brick is having issues or when you want to grow the capacity of a volume without adding bricks.

Bricks can be replaced while the volume they are a part of is on-line. Use the following command to initiate a replace operation: **gluster volume replace-brick <VOLUME> <OLD-BRICK> <NEW-BRICK> start**.

You can view the progress of a replacement operation after it has been started using the command: **gluster volume replace-brick <VOLUME> <OLD-BRICK> <NEW-BRICK> status**.

Once the status indicates that the replacement operation is complete, you will need to *commit* the operation. Before the commit both the old and the new brick will be used, committing will remove the old brick from your volume: **gluster volume replace-brick <VOLUME> <OLD-BRICK> <NEW-BRICK> commit**



References

Red Hat Storage Server Administration Guide

- Section 10.4: Migrating Volumes
- Chapter 21: Troubleshooting



Performance Checklist

Investigate Self-Heal

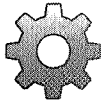
Lab Overview: In this exercise you will investigate the *Self-Heal* feature of Red Hat Storage Server.

Before you begin...

Make sure you still have a working **replvol** volume mounted on **/mnt/replvol** on your **cXn5** machine using a floating IP address.

- ☐ 1. Use your **cXn3** machine to investigate which bricks make up your **replvol** volume.
- ☐ 2. Power down your **cXn3** machine.
- ☐ 3. On your **cXn5** machine create 10 new 1 MiB files called **/mnt/replvol/testfile1** through **/mnt/replvol/testfile10**.
- ☐ 4. On your **cXn4** machine inspect the contents of **/bricks/brick4** and **/bricks/brick12**. You should see the various **testfile*** files distributed (somewhat) evenly between the two.
- ☐ 5. Power on your **cXn3** machine.
- ☐ 6. Once **cXn3** is back on-line, log in as **root** and inspect the contents of **/bricks/brick3** and **/bricks/brick11**. What do you notice?
- ☐ 7. Request a list of files on your **replvol** volume that still need to be healed.
- ☐ 8. Wait for the self-heal daemon to run, or trigger a self heal for your **replvol** volume if you are impatient.
- ☐ 9. Request a list of files that have been healed on the **replvol** volume.
- ☐ 10. Check if any files failed the self-heal.
- ☐ 11. Check the contents of **/bricks/brick3** and **/bricks/brick11** on your **cXn3** machine. Do the files now have the correct size?

Unit Test



Case Study

Replacing a Brick

Lab Overview: In this exercise you will replace a brick in your **replvol** volume without causing any downtime.

Success Criteria: You will have successfully completed this lab when you have successfully replaced a brick in your **replvol** volume using the criteria below.

Before you begin...

Make sure you still have a working **replvol** volume.

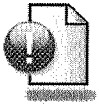
The storage backing **/bricks/brick3** on your **cXn3** machine has been acting up lately. You have been asked to replace that brick with a new 2 GiB brick mounted on **/bricks/brick17** on your **cXn1** machine without causing any downtime or service disruption. It should be backed by a logical volume called **/dev/vg_bricks/brick17**.

After the replacement is complete, remove the old logical volume **/dev/vg_bricks/brick3** on your **cXn3** machine.

How would you address the case study described above? Take notes on your process in the space below and then implement it.



Personal Notes



Unit Summary

Troubleshooting

- Locate Logs
- Use Self-Heal
- Replace Faulty Bricks



APPENDIX A

SOLUTIONS

Introduction to Red Hat Storage Server



Multiple Choice Quiz

Red Hat Storage Server Features

Please answer the following questions:

1. Red Hat Storage Server can be deployed to...
(select one or more of the following...)
 - a. Physical Hardware
 - b. Public Cloud
 - c. Private Cloud
 - d. Hybrid Cloud

2. Red Hat Storage Server uses a central metadata node to locate files.
(select one of the following...)
 - a. True
 - b. False

3. Red Hat Storage Server is built around the following technology:
(select one of the following...)
 - a. Ceph
 - b. DRBD
 - c. GFS2
 - d. GlusterFS



Multiple Choice Quiz

Concepts and Terminology

Please answer the following questions:

1. In order to work together, multiple Red Hat Storage Servers **must** be combined into a Trusted Storage Pool.
(select one of the following...)
 - a. True
 - b. False

2. When retrieving or storing a file on a distributed volume its location is determined by:
(select one of the following...)
 - a. Querying a central index

- b. Querying the other nodes in the Trusted Storage Pool
- c. /dev/urandom
- d. **An Elastic Hash Algorithm**

3. A brick can belong to multiple volumes at a time.

(select one of the following...)

- a. True
- b. **False**

Explore the Classroom Environment



Performance Checklist

Explore the Classroom Environment

Lab Overview: In this exercise you will explore the classroom network and the machines available for your use.

- ❑ 1. If you haven't already done so log in to your **desktopX** machine as the user **student** with the password **student**.

If you are taking this course in a physical classroom your **desktopX** machine is the physical machine at which you are seated.

If you are taking this course in a virtual environment you can access your **desktopX** machine by locating the entry for **desktopX** in the left-hand menu and double-clicking on it to open the console. If your machine has not yet started, use the **Start** button above the console to start it.

- ❑ 2. Open a terminal on your **desktopX** machine.

Either right-click on your desktop and select **Open in Terminal** or select **Applications > System Tools > Terminal**.

- ❑ 3. Inspect the hostname for your **desktopX** machine. This should be something like **desktopX.example.com**. Note the hostname and the **X** part in the table below for future reference.

```
[student@desktopX ~]$ hostname
desktopX.example.com
```

- ❑ 4. Inspect the IP address assigned to your **desktopX** machine. The address for this machine will be assigned to the **br0** interface.

```
[student@desktopX ~]$ ip a s <eth0|br0>
4: br0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN
    link/ether 00:10:18:2b:98:6b brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.X0/24 brd 192.168.0.255 scope global br0
    inet6 fe80::210:18ff:fe2b:986b/64 scope link
        valid_lft forever preferred_lft forever
```

Setting	Value
Hostname	
X	
IP Address	



Performance Checklist

Accessing your Virtual Machines

Lab Overview: In this exercise you will practice accessing your virtual machines.

- ❑ 1. Reset your **node1** virtual machines. If you are in a physical classroom this can be done by executing the command **lab-setup-nodes -1** as **root** from your **desktopX** machine. In a virtual learning environment open the console for your **node1** virtual machine and press the **Reset** button located above the console.

In a *physical classroom* open a terminal on your desktopX machine and execute the following commands:

```
[student@desktopX ~]$ su -
Password: redhat
[root@desktopX ~]# lab-setup-nodes -1
```

If this is the first time that you are resetting a virtual machine the **lab-setup-nodes** command will download a base virtual machine from the classroom instructor system first.

- ❑ 2. Open a console to your **node1/cXn1** virtual machine, and log in as **root** with the password **redhat**.

In a physical classroom you can do this by executing the command **virt-viewer node1** as **root** or by double-clicking on the **node1** virtual machine in **virt-manager** (Applications > System Tools > Virtual Machine Manager).

In a *virtual classroom* you can do this by double-clicking on on the **node1/cXn1** virtual machine in the left-hand menu. This button will only be available if the machine is currently powered down, and after resetting you will have to use the **Power On** button to start your machine.

- ❑ 3. Determine the hostname of your virtual machine. This should be **cXn1.example.com**, where **X** is the number you determined in the previous exercise.

```
[root@cXn1.example.com ~]# hostname
cXn1.example.com
```

- ❑ 4. Determine the IP address for your virtual machine. This IP address should be **192.168.0.X1**, where **X** is your desktop number, and it should be assigned to the **eth0** interface.

```
[root@cXn1.example.com ~]# ip a s eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 52:54:00:00:0d:01 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.X1/24 brd 192.168.0.255 scope global eth0
    inet6 fe80::5054:ff:fe00:d01/64 scope link
        valid_lft forever preferred_lft forever
```

- ❑ 5. Open a terminal as the user **student** on your **desktopX** machine and create a new **SSH key-pair**. Do not set a passphrase on this key-pair for classroom use.

```
[student@desktopX ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/student/.ssh/id_rsa): Enter
Enter passphrase (empty for no passphrase): Enter
Enter same passphrase again: Enter
Your identification has been saved in /home/student/.ssh/id_rsa.
Your public key has been saved in /home/student/.ssh/id_rsa.pub.
The key fingerprint is:
43:fa:9a:a9:96:91:de:d5:a7:d4:7b:ef:99:e2:ad:69 student@desktopX.example.com
The key's randomart image is:
+--[ RSA 2048]-----+
|
|
|      .
|     o
|    .. S. .
|   o ...o o
|  . + ... o .
|  + .+ . .E+ o
| ...+      o=o*o
+-----+
```

- ❑ 6. Copy the public part of your newly generated key-pair to the **root** account on your **cXn1** machine so you can use **SSH** without entering a password.

```
[student@desktopX ~]$ ssh-copy-id root@cXn1.example.com
The authenticity of host 'cXn1.example.com (192.168.0.X1)' can't be established.
RSA key fingerprint is 9d:8e:8f:db:07:36:7e:54:ec:0b:c1:ce:01:7d:77:f1.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'cXn1.example.com,192.168.0.X1' (RSA) to the list of
known hosts.
root@cXn1.example.com's password: redhat
Now try logging into the machine, with "ssh 'root@cXn1.example.com'", and check
in:

    .ssh/authorized_keys

to make sure we haven't added extra keys that you weren't expecting.
```

- 7. To make your life easier, you can configure **SSH** to always connect to the **root** account when you **ssh** into your virtual machines. To do this create a new file **/home/student/.ssh/config** on **desktopX** with the following content. Make sure to replace **X** with your desktop number.

```
Host cXn* cXn*.example.com
User root
```

Set the permissions on that file to **0600**.

```
[student@desktopX ~]$ cat > .ssh/config << EOF
Host cXn* cXn*.example.com
  User root
EOF
```

```
[student@desktopX ~]$ chmod 0600 .ssh/config
```

- 8. As **student** on your **desktopX** machine, attempt to **ssh** to **cXn1.example.com**. This should log you in automatically as **root**.

```
[student@desktopX ~]$ ssh cXn1.example.com
Last login: Wed Nov 14 12:36:50 2012 from desktopX.example.com
[root@cXn1 ~]#
```

- 9. Determine the default runlevel of your **cXn1** machine.

```
[root@cXn1 ~]# grep initdefault /etc/inittab
id:3:initdefault:
```

- 10. Determine which services are started in the default runlevel.

```
[root@cXn1 ~]# chkconfig --list | grep 3:on
abrt-ccpp      0:off 1:off 2:off 3:on  4:off 5:on  6:off
abrt-oops      0:off 1:off 2:off 3:on  4:off 5:on  6:off
abrt-d         0:off 1:off 2:off 3:on  4:off 5:on  6:off
...output truncated...

fixnetwork     0:off 1:off 2:off 3:on  4:off 5:off 6:off
gluster-system-settings 0:off 1:off 2:on  3:on  4:on  5:on  6:off
glusterd       0:off 1:off 2:off 3:on  4:off 5:on  6:off
...output truncated...
```

Note: While the **gluster-system-settings** and **glusterd** services are enabled by default on any install of Red Hat Storage Server, the **fixnetwork** service is specific to our classroom installation.



Note

If you want to use password-less login with SSH key-pairs, copy your public key to every machine you want to access in this manner.

When you reset a virtual machine it will be brought back to a pristine state, and as such you will have to copy your keys again.

Installing Red Hat Storage Server



Performance Checklist

Installing Red Hat Storage Server

Lab Overview: In this exercise you will install a new virtual machine with Red Hat Storage Server.

Before you begin...

Some instructions in this exercise differ based on whether you are taking this course in a physical classroom or in a virtual environment. Pay close attention to the instructions in the different steps to see if those steps apply to your environment.

- ❑ 1. Reset your **node1** virtual machine, then when the machine is starting press **Ctrl+B** when you see the **gPXE** prompt.

At the **gPXE** prompt type **autoboot** to start a PXE installation.

Sometime **autoboot** fails the first time. If this happens to you try again.

Physical Classroom Only: To reset your **node1** virtual machine run the following command as root on your **desktopX** machine:

```
[root@desktopX ~]# lab-setup-nodes -1
```

If this is the first time that you are resetting a virtual a base image will be downloaded from the instructor server.

Virtual Classroom Only: Power down your **node1** virtual machine, then power it back on. When you see the **gPXE** prompt press **Ctrl+B**, then type **autoboot**. You will have to click inside your console window first before keypresses are registered. Do *not* use the **install** button, as it will simply power on your machine.

- ❑ 2. When the PXE boot menu appears use your cursor keys to select **Red Hat Storage Server**, then press **Enter**.

- ❑ 3. Enter **redhat** in both the **Root Password** and **Confirm** fields, then click on **Next**.

Physical classroom only: If you do not see the **Next** button select **View > Resize to VM**.

- ❑ 4. Click on the **Use Anyway** button when you are informed of your weak password choice.
- ❑ 5. Select **Use All Space** and **Review and modify partitioning layout**, then click on **Next**.
- ❑ 6. Inspect the default partitioning selected for you, then click on **Next**.

Note: If you are performing an installation outside of the classroom you will want to pay close attention to this screen. Make sure the installer is not allocating the drives/space that you intended for your bricks to the root file system.

- ❑ 7. Accept that your drives will be wiped irrevocably by clicking on the **Format** button.

- ☐ 8. Sit back, relax, and enjoy the beautiful progress bar that now graces your screen. This installation should only take a few moments.
- ☐ 9. Once the installer has finished, click the **Reboot** button and wait for the system to shut down. (**Virtual Machine Manager** does not reboot a machine in installation mode, instead it shuts it down.) Click the **Power on the virtual machine** button (shaped like a *Play* icon).
- ☐ 10. Wait for the machine to start then login as **root** using the password you specified at the beginning of the installation process (**redhat**).
- ☐ 11. Verify the **glusterd** service has been automatically started.

```
[root@localhost ~]# service glusterd status
glusterd (pid 1353) is running...
```

- ☐ 12. Shut down your virtual machine.

```
[root@localhost ~]# poweroff
```

Basic Configuration



Performance Checklist

Build a Volume

Lab Overview: In this exercise you will create your first Red Hat Storage Server cluster and your first volume.

Success Criteria: You will have successfully completed this lab when you have a new, running volume called **firstvol**.

Lab Outline In this exercise you will build a two-node Red Hat Storage Server cluster, then create storage bricks and a storage volume.

- ❑ 1. Reset your **node1** and **node2** virtual machines.

In a physical classroom, this can be done by executing the **lab-setup-nodes -12** command as **root** from your **desktopX** machine.

In a virtual learning environment, open the consoles and press the **Reset** button above the consoles for your **node1** and **node2** virtual machines. This button will only be available if the machines are currently powered down, and you will have to manually power on those machine using the **Power On** button afterwards.

In a physical classroom, open a terminal on your desktopX machine and execute the following commands:

```
[student@desktopX ~]$ su -  
Password: redhat  
[root@desktopX ~]# lab-setup-nodes -12
```

If this is the first time you are resetting a virtual machine, the **lab-setup-nodes** command will download a base virtual machine from the classroom instructor system first.

- ❑ 2. Log into your **cXn1.example.com** system as **root** and peer with your **cXn2.example.com** system. Remember that **X** is your desktop number.

```
[root@cXn1 ~]# gluster peer probe cXn2.example.com  
Probe successful
```

- ❑ 3. Verify the systems have successfully peered from your **cXn1** system.

```
[root@cXn1 ~]# gluster peer status  
Number of peers: 1  
  
Hostname: cXn2.example.com  
Uuid: 1d8507b1-cdf2-4723-b412-a2c0ae979a4e  
State: Peer in Cluster (Connected)
```

- ❑ 4. On your **cXn1** system create a new 2 GiB logical volume in the **vg_bricks** volume group called **brick1**. The **vg_bricks** volume group has already been created for you.


```
[root@cXn1 ~]# lvcreate -L 2G -n brick1 vg_bricks
```

- 5. On your **cXn2** system create a new 2 GiB logical volume in the **vg_bricks** volume group called **brick2**. The **vg_bricks** volume group has already been created for you.

```
[root@cXn2 ~]# lvcreate -L 2G -n brick2 vg_bricks
```

- 6. Create an XFS file system on both of your new logical volumes. Be sure you set the inode size to 512 bytes.

```
[root@cXn1 ~]# mkfs.xfs -i size=512 /dev/vg_bricks/brick1
meta-data=/dev/vg_bricks/brick1 isize=512    agcount=4, agsize=131072 blks
        =                               sectsz=512    attr=2
data                =                               bsize=4096    blocks=524288, imaxpct=25
        =                               sunit=0      swidth=0 blks
naming              =version 2                  bsize=4096    ascii-ci=0
log                 =internal log                bsize=4096    blocks=2560, version=2
        =                               sectsz=512    sunit=0 blks, lazy-count=1
realtime            =none                       extsz=4096    blocks=0, rtextents=0
```

```
[root@cXn2 ~]# mkfs.xfs -i size=512 /dev/vg_bricks/brick2
meta-data=/dev/vg_bricks/brick2 isize=512    agcount=4, agsize=131072 blks
        =                               sectsz=512    attr=2
data                =                               bsize=4096    blocks=524288, imaxpct=25
        =                               sunit=0      swidth=0 blks
naming              =version 2                  bsize=4096    ascii-ci=0
log                 =internal log                bsize=4096    blocks=2560, version=2
        =                               sectsz=512    sunit=0 blks, lazy-count=1
realtime            =none                       extsz=4096    blocks=0, rtextents=0
```

- 7. On your **cXn1** and **cXn2** machines respectively, create the **/bricks/brick1** and **/bricks/brick2** mount points.

```
[root@cXn1 ~]# mkdir -p /bricks/brick1
```

```
[root@cXn2 ~]# mkdir -p /bricks/brick2
```

- 8. On both your nodes *persistently* mount your new file systems on the mount points you just created.

On both your nodes add the following line to **/etc/fstab**, replacing **Y** with the node number (**1** or **2**).

```
/dev/vg_bricks/brickY /bricks/brickY xfs defaults 0 2
```

Execute the following command on both your nodes:

```
[root@cXnY ~]# mount -a
```

- 9. From your **cXn1** machine create a new volume called **firstvol**, using the two bricks you have just created. Do not specify any other options.

```
[root@cXn1 ~]# gluster volume create firstvol \  
> cXn1.example.com:/bricks/brick1 \  
> cXn2.example.com:/bricks/brick2  
Creation of volume firstvol has been successful. Please start the volume to access  
data.
```

- 10. From your **cXn1** machine start your new volume.

```
[root@cXn1 ~]# gluster volume start firstvol  
Starting volume firstvol has been successful
```

Volume Types



Performance Checklist

Create a Replicated Volume

Lab Overview: In this exercise you will create a replicated volume using two 2 GiB bricks located on **cXn3** and **cXn4**.

Success Criteria: You will have successfully completed this lab when you have a new, running volume called **replvol** consisting of two bricks.

Lab Outline In this exercise you will add two more nodes to your cluster, add bricks to your new nodes, then create a new replicated volume.

Before you begin...

Make sure you still have a working two-node cluster on your **cXn1** and **cXn2** machines.

- ❑ 1. Begin by resetting your **node3** and **node4** virtual machines.

Students in a physical classroom can do this by executing the **lab-setup-nodes -34** command as **root** on their **desktopX** machine.

Students in a virtual training environment must open the consoles for their **node3** and **node4** machines and click the **Reset** button above those consoles. Remember that this button will only be available when a machine is currently powered off, and you will have to manually start those machines using the **Power On** button afterwards.

In a physical classroom open a terminal on your **desktopX** machine and execute the following commands:

```
[student@desktopX ~]$ su -
Password: redhat
[root@desktopX ~]# lab-setup-nodes -34
```

If this is the first time that you are resetting a virtual machine, the **lab-setup-nodes** command will download a base virtual machine from the classroom instructor system first.

- ❑ 2. From your **cXn1** machine, add your **cXn3.example.com** and **cXn4.example.com** machines to your cluster.

```
[root@cXn1 ~]# gluster peer probe cXn3.example.com
Probe successful
[root@cXn1 ~]# gluster peer probe cXn4.example.com
Probe successful
```

- ❑ 3. Confirm the nodes have been added to your cluster.

```
[root@cXn1 ~]# gluster peer status
Number of Peers: 3

Hostname: cXn2.example.com
Uuid: 1d8507b1-cdf2-4723-b412-a2c0ae979a4e
```

```

State: Peer in Cluster (Connected)

Hostname: cXn3.example.com
Uuid: 1c615aff-5b0f-4d1c-b999-5d8badbf0b26
State: Peer in Cluster (Connected)

Hostname: cXn4.example.com
Uuid: 8d88e4b1-a52e-4606-bdc6-e7d4a7298d83
State: Peer in Cluster (Connected)

```

- 4. On both of your new nodes create a 2 GiB logical volume in the **vg_bricks** volume group called **brickY**, where **Y** is the node number. Format it with an XFS file system with 512-byte inodes and persistently mount it on **/bricks/brickY**.

Repeat the commands below on both of your new nodes. Make sure to replace every occurrence of **Y** with the correct node number (3 or 4).

```

[root@cXnY ~]# lvcreate -L 2G -n brickY vg_bricks
Logical volume "brickY" created
[root@cXnY ~]# mkfs.xfs -i size=512 /dev/vg_bricks/brickY
meta-data=/dev/vg_bricks/brickY isize=512    agcount=4, agsize=131072 blks
=                               sectsz=512   attr=2
data      =                       bsize=4096   blocks=524288, imaxpct=25
=                               sunit=0      swidth=0 blks
naming    =version 2              bsize=4096   ascii-ci=0
log       =internal log          bsize=4096   blocks=2560, version=2
=                               sectsz=512   sunit=0 blks, lazy-count=1
realtime  =none                  extsz=4096   blocks=0, rtextents=0
[root@cXnY ~]# mkdir -p /bricks/brickY
[root@cXnY ~]# echo "/dev/vg_bricks/brickY /bricks/brickY xfs defaults 0 2" >> /
etc/fstab
[root@cXnY ~]# mount -a

```

- 5. Using the two new bricks you just created, create a new *Replicated* volume called **replvol**.

```

[root@cXn1 ~]# gluster volume create replvol replica 2 \
> cXn3.example.com:/bricks/brick3 \
> cXn4.example.com:/bricks/brick4
Creation of volume replvol has been successful. Please start the volume to access
data.

```

- 6. Inspect the properties of your new volume.

```

[root@cXn1 ~]# gluster volume info replvol

Volume Name: replvol
Type: Replicate
Volume ID: 1f32b6e6-a409-4d5f-aa88-793c4a770253
Status: Created
Number of Bricks: 1 x 2 = 2
Transport-type: tcp
Bricks:
Brick1: cXn3.example.com:/bricks/brick3
Brick2: cXn4.example.com:/bricks/brick4

```

- 7. Start your new volume.

```
[root@cXn1 ~]# gluster volume start replvol
Starting volume replvol has been successful
```



Case Study

Creating a Distributed-Replicated Volume

Lab Overview: In this exercise you will create a distributed-replicated volume using four bricks.

Success Criteria: You have successfully completed this lab when you have a new, running distributed-replicated volume called **distreplvol** using four bricks.

Lab Outline In this lab you will create the bricks necessary for a four brick distributed-replicated volume, then create and start that volume.

Before you begin...

Make sure you still have a working four-node cluster.

You have been asked to create a new **2 × 2 distributed-replicated** volume using four 2 GiB bricks. The volume should be called **distreplvol**.

The four bricks should be mounted on **/bricks/brick5** through **/bricks/brick8**, and their logical volume names should be **/dev/vg_bricks/brick5** through **/dev/vg_bricks/brick8** respectively. The bricks should be distributed in such a way that **brick5** resides on **cXn1.example.com**, **brick6** on **cXn2.example.com**, etc.

1. Prepare your bricks. Execute the following commands on all four of your nodes, replacing **Y** with the node number, and **Y+4** by the the node number with 4 added to it. e.g **Y+4** becomes 7 when **Y** is 3.

```
[root@cXnY ~]# lvcreate -L 26 -n brickY+4 vg_bricks
Logical volume "brickY+4" created
[root@cXnY ~]# mkfs.xfs -i size=512 /dev/vg_bricks/brickY+4
meta-data=/dev/vg_bricks/brickY+4 isize=512    agcount=4, agsize=131072 blks
       =                               sectsz=512   attr=2
data     =                               bsize=4096   blocks=524288, imaxpct=25
       =                               sunit=0      swidth=0 blks
naming   =version 2                   bsize=4096   ascii-ci=0
log      =internal log                bsize=4096   blocks=2560, version=2
       =                               sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                       extsz=4096   blocks=0, rtextents=0
[root@cXnY ~]# mkdir -p /bricks/brickY+4
[root@cXnY ~]# echo "/dev/vg_bricks/brickY+4 /bricks/brickY+4 xfs defaults 0 2" >> /
etc/fstab
[root@cXnY ~]# mount -a
```

2. Create a new distributed-replicated volume called **distreplvol** using the bricks you just created. Since we are using four bricks and want the data to be both distributed and replicated, replicate all data across two bricks.

```
[root@cXn1 ~]# gluster volume create distreplvol replica 2 \
> cXn1.example.com:/bricks/brick5 \
> cXn2.example.com:/bricks/brick6 \
```

```
> cXn3.example.com:/bricks/brick7 \  
> cXn4.example.com:/bricks/brick8  
Creation of volume distreplvol has been successful. Please start the volume to access  
data.
```

3. Inspect your new volume to confirm that is built correctly.

```
[root@cXn1 ~]# gluster volume info distreplvol  
  
Volume Name: distreplvol  
Type: Distributed-Replicate  
Volume ID: 55d75745-68e1-4b00-8f29-c7287524e730  
Status: Created  
Number of Bricks: 2 x 2 = 4  
Transport-type: tcp  
Bricks:  
Brick1: cXn1.example.com:/bricks/brick5  
Brick2: cXn2.example.com:/bricks/brick6  
Brick3: cXn3.example.com:/bricks/brick7  
Brick4: cXn4.example.com:/bricks/brick8
```

4. Start your new volume.

```
[root@cXn1 ~]# gluster volume start distreplvol  
Starting volume distreplvol has been successful
```

Clients



Performance Checklist

Using the Native Client

Lab Overview: In this exercise you will configure your **cXn5.example.com** machine to persistently mount your **firstvol** volume on **/mnt/firstvol** using the native client.

Success Criteria: You will have successfully completed this lab when you have mounted your **firstvol** volume using the native client.

Before you begin...

Make sure you still have a running **firstvol** volume on your cluster.

- ☐ 1. Reset your **node5** virtual machine.

In a physical classroom this can be done by executing the **lab-setup-nodes -5** command as **root** from the desktopX machine.

In a virtual learning environment, open the console for the **node5** virtual machine and click the **Reset** button located above the console. This button will only be available if the machine is currently powered down, and you have to manually start the machine using the **Power On** button afterwards.

In a physical classroom open a terminal on your desktopX machine and execute the following commands:

```
[student@desktopX ~]$ su -  
Password: redhat  
[root@desktopX ~]# lab-setup-nodes -5
```

If this is the first time that you are resetting a virtual machine, the **lab-setup-nodes** command will download a base virtual machine from the classroom instructor system first.

- ☐ 2. On your **cXn5.example.com** machine create a mount point called **/mnt/firstvol**.

```
[root@cXn5 ~]# mkdir -p /mnt/firstvol
```

- ☐ 3. On your **cXn5.example.com** machine add an **/etc/fstab** entry to automatically mount the **firstvol** volume on **/mnt/firstvol** using the native client. Remember to add the **_netdev** mount option to make sure the mount will work at boot time.

Add a line like the following to **/etc/fstab** on your **cXn5.example.com** machine:

```
cXn1.example.com:/firstvol /mnt/firstvol glusterfs _netdev 0 0
```

- ☐ 4. Activate your new mount on **cXn5**.

```
[root@cXn5 ~]# mount -a
```

- ❑ 5. On **cXn5** create 10 new files called **file1** through **file10** in **/mnt/firstvol**.

```
[root@cXn5 ~]# touch /mnt/firstvol/file{1..10}
```

- ❑ 6. On **cXn1** and **cXn2** inspect the contents of the bricks that make up your **firstvol** volume (**/bricks/brick1** and **/bricks/brick2** respectively).

What do you notice about the file distribution?

```
[root@cXn1 ~]# ls -l /bricks/brick1
```

```
[root@cXn2 ~]# ls -l /bricks/brick2
```

The ten files should be distributed (somewhat) evenly across the two bricks.



Performance Checklist

Using NFS Clients

Lab Overview: In this exercise you will mount your **replvol** volume over NFSv3 using TCP.

Success Criteria: You will have successfully completed this lab when your **replvol** volume is persistently mounted on **/mnt/replvol** on **cXn5** using NFS.

Before you begin...

Make sure your **cXn5** machine is still operational and your **replvol** volume is still running.

- ❑ 1. On your **cXn5.example.com** machine create a mount point called **/mnt/replvol**.

```
[root@cXn5 ~]# mkdir -p /mnt/replvol
```

- ❑ 2. On your **cXn5.example.com** machine create an **/etc/fstab** entry to automatically mount your **replvol** volume on **/mnt/replvol** using NFSv3.

Add a line like the following to your **/etc/fstab** on **cXn5.example.com**:

```
cXn1.example.com:/replvol /mnt/replvol nfs vers=3 0 0
```

- ❑ 3. Activate your new mount on **cXn5**.

```
[root@cXn5 ~]# mount -a
```

- ❑ 4. On **cXn5** create 10 new files called **file1** through **file10** in **/mnt/replvol**.

```
[root@cXn5 ~]# touch /mnt/replvol/file{1..10}
```

- ❑ 5. On **cXn3** and **cXn4** inspect the contents of the bricks that make up your **replvol** volume (**/bricks/brick3** and **/bricks/brick4** respectively).

What do you notice about the file distribution?

```
[root@cXn3 ~]# ls -l /bricks/brick3
```

```
[root@cXn4 ~]# ls -l /bricks/brick4
```

The ten files should be present on both bricks.



Performance Checklist

Configuring CIFS Clients

Lab Overview: In this exercise you will add a Samba-only user to your **cXn1** machine then mount your **distreplvol** volume on your **desktopX** machine using the CIFS protocol.

Success Criteria: You have successfully completed this lab when you have mounted your **distreplvol** on your **desktopX** machine using CIFS.

Before you begin...

Make sure you still have a running volume called **distreplvol**.

- ❑ 1. On your **cXn1** machine start and enable the **smb** service.

```
[root@cXn1 ~]# chkconfig smb on
[root@cXn1 ~]# service smb start
```

- ❑ 2. On your **cXn1** machine create a samba-only user called **cifsuser** with a Samba password of **redhat**. (Samba-only means the user should not be able to log into your machine using any other protocol than CIFS.)

```
[root@cXn1 ~]# useradd -s /sbin/nologin cifsuser
[root@cXn1 ~]# smbpasswd -a cifsuser
New SMB password: redhat
Retype new SMB password: redhat
Added user cifsuser.
```

- ❑ 3. Give your new user write permissions on the **distreplvol** volume. One of the ways to do this is to change the permissions on the automatic mount done on **/mnt/samba/distreplvol** on any of your storage nodes.

```
[root@cXn1 ~]# chown :cifsuser /mnt/samba/distreplvol/.
[root@cXn1 ~]# chmod 775 /mnt/samba/distreplvol/.
```

- ❑ 4. On your **desktopX** system create a mount point called **/mnt/distreplvol**.

```
[root@desktopX ~]# mkdir -p /mnt/distreplvol
```

- ❑ 5. On your **desktopX** machine *temporarily* mount your **distreplvol** volume on **/mnt/distreplvol** using the CIFS protocol. For authentication you can use the **cifsuser** user you have just created with the password **redhat**.

```
[root@desktopX ~]# mount -t cifs -o user=cifsuser,password=redhat //
cXn1.example.com/gluster-distreplvol /mnt/distreplvol
```

- ❑ 6. On **desktopX** create 10 new files called **file1** through **file10** in **/mnt/distreplvol**.

```
[root@desktopX ~]# touch /mnt/distreplvol/file{1..10}
```

- ❑ 7. On **cXn1** through **cXn4** inspect the contents of the bricks that make up your **distreplvol** volume (**/bricks/brick5** through **/bricks/brick8** respectively).

What do you notice about the file distribution?

```
[root@cXn1 ~]# ls -l /bricks/brick5
```

```
[root@cXn2 ~]# ls -l /bricks/brick6
```

```
[root@cXn3 ~]# ls -l /bricks/brick7
```

```
[root@cXn4 ~]# ls -l /bricks/brick8
```

brick5 and **brick6** are mirrors of each other, just like **brick7** and **brick8**. Your files should be distributed across these two mirrors.

- ❑ 8. Unmount your **distreplvol** volume from **desktopX**.

```
[root@desktopX ~]# umount /mnt/distreplvol
```



Performance Checklist

Setting Volume Options

Lab Overview: In this exercise you will set, and reset, volume options for your **firstvol** volume.

Before you begin...

Make sure you still have a working **replvol** volume that is persistently mounted on **/mnt/replvol** on your **cXn5** machine.

- ❑ 1. *Temporarily* unmount your **replvol** volume from your **cXn5** machine.

```
[root@cXn5 ~]# umount /mnt/replvol
```

- ❑ 2. Set the **nfs.disable** option to **On** for your **replvol** volume.

```
[root@cXn1 ~]# gluster volume set replvol nfs.disable On
Set volume successful
```

- 3. Inspect the volume options for your **replvol** volume.

```
[root@cXn1 ~]# gluster volume info replvol

Volume Name: replvol
Type: Replicate
Volume ID: 1f32b6e6-a409-4d5f-aa88-793c4a770253
Status: Started
Number of Bricks: 1 x 2 = 2
Transport-type: tcp
Bricks:
Brick1: cXn3.example.com:/bricks/brick3
Brick2: cXn4.example.com:/bricks/brick4
Options Reconfigured:
nfs.disable: On
```

- 4. On your **cXn5** machine, attempt to mount **/mnt/replvol** again. This should fail.

```
[root@cXn5 ~]# mount /mnt/replvol
mount.nfs: mounting cXn1.example.com:/replvol failed, reason given by server:
No such file or directory
```

- 5. Reset the **nfs.disable** option for your **replvol** volume back to its default setting.

```
[root@cXn1 ~]# gluster volume reset replvol nfs.disable
reset volume successful
```

- 6. Display the volume options for your **replvol** volume.

```
[root@cXn1 ~]# gluster volume info replvol

Volume Name: replvol
Type: Replicate
Volume ID: 1f32b6e6-a409-4d5f-aa88-793c4a770253
Status: Started
Number of Bricks: 1 x 2 = 2
Transport-type: tcp
Bricks:
Brick1: cXn3.example.com:/bricks/brick3
Brick2: cXn4.example.com:/bricks/brick4
```

- 7. Attempt to mount **/mnt/replvol** on your **cXn5** machine again. This time it should work.

```
[root@cXn5 ~]# mount /mnt/replvol
```



Case Study

Limit Access to Volumes

Lab Overview: In this exercise you will limit access to your **replvol** volume.

Success Criteria: You will have successfully completed this lab when only clients in the **192.168.0.0/24** subnet can connect to your **replvol** volume, with the exception of your **desktopX** machine.

Before you begin...

Make sure you still have a working **replvol** volume.

You have been asked to limit access to your **replvol** volume to only clients in the **192.168.0.0/24** subnet.

Note: Reconfiguring access restrictions requires stopping and starting the volume.

1. Begin by *temporarily* unmounting your **replvol** volume from your **cXn5** machine.

```
[root@cXn5 ~]# umount /mnt/replvol
```

2. Set the option **auth.allow** for your **replvol** volume to **192.168.0.***.

```
[root@cXn1 ~]# gluster volume set replvol auth.allow "192.168.0.*"
Set volume successful
```

3. Inspect the volume options for your **replvol** volume.

```
[root@cXn1 ~]# gluster volume info replvol

Volume Name: replvol
Type: Replicate
Volume ID: 1f32b6e6-a409-4d5f-aa88-793c4a770253
Status: Started
Number of Bricks: 1 x 2 = 2
Transport-type: tcp
Bricks:
Brick1: cXn3.example.com:/bricks/brick3
Brick2: cXn4.example.com:/bricks/brick4
Options Reconfigured:
auth.allow: 192.168.0.*
```

4. Stop then start your **replvol** volume.

```
[root@cXn1 ~]# gluster volume stop replvol
Stopping volume will make its data inaccessible. Do you want to continue? (y/n) y
Stopping volume replvol has been successful
[root@cXn1 ~]# gluster volume start replvol
Starting volume replvol has been successful
```

5. Mount your **replvol** volume on your **cXn5** machine again.

```
[root@cXn5 ~]# mount /mnt/replvol
```

ACLs and Quotas



Performance Checklist

Using POSIX ACLs

Lab Overview: In this exercise you will reconfigure your mount of **firstvol** on your **cXn5** machine to use ACLs.

Success Criteria: You will have successfully completed this lab when you have implemented POSIX ACLs on your **firstvol** volume.

Before you begin...

Make sure you still have your **firstvol** volume mounted on **/mnt/firstvol** on your **cXn5** machine using the native client.

- ❑ 1. Add the **acl** mount option to **/etc/fstab** on **cXn5** for the **firstvol** volume.

Your entry in **/etc/fstab** should now look something like this:

```
cXn1.example.com:/firstvol /mnt/firstvol glusterfs _netdev,acl 0 0
```

- ❑ 2. Re-mount your **firstvol** volume on **cXn5**.

Note: Since the native client currently does not support the **remount** mount option, you will have to unmount then mount your volume.

```
[root@cXn5 ~]# umount /mnt/firstvol
[root@cXn5 ~]# mount /mnt/firstvol
```

- ❑ 3. Create a new user on **cXn5** named **acluser**.

```
[root@cXn5 ~]# useradd acluser
```

- ❑ 4. Create a new directory on **cXn5** named **/mnt/firstvol/supersecret** that is owned by the user **root** and the group **root**, and has permissions set to **0700**.

```
[root@cXn5 ~]# mkdir -m 0700 /mnt/firstvol/supersecret
```

- ❑ 5. As the **acluser** user on **cXn5**, try to list the contents of the **/mnt/firstvol/supersecret** directory. This should fail.

```
[root@cXn5 ~]# su - acluser -c 'ls -la /mnt/firstvol/supersecret '
ls: cannot open directory /mnt/firstvol/supersecret: Permission denied
```

- ❑ 6. Add an ACL to the **/mnt/firstvol/supersecret** directory allowing **acluser** full access (read, write, and execute).

```
[root@cXn5 ~]# setfacl -m u:acluser:rwX /mnt/firstvol/supersecret
```

- ❑ 7. Also add an ACL to **/mnt/firstvol/supersecret** allowing **acluser** full access to any new files and directories created underneath it.

```
[root@cXn5 ~]# setfacl -m d:u:acluser:rwX /mnt/firstvol/supersecret
```

- ❑ 8. Verify the ACLs you have just created with **getfacl**.

```
[root@cXn5 ~]# getfacl /mnt/firstvol/supersecret
getfacl: Removing leading '/' from absolute path names
# file: mnt/firstvol/supersecret/
# owner: root
# group: root
user::rwX
user:acluser:rwX
group:---
mask::rwX
other:---
default:user::rwX
default:user:acluser:rwX
default:group:---
default:mask::rwX
default:other:---
```

- ❑ 9. Attempt to create a new file called **/mnt/firstvol/supersecret/launchcodes.txt** as **acluser** then verify the ACLs on the newly created file.

```
[root@cXn5 ~]# su - acluser -c 'echo 00000000 > /mnt/firstvol/supersecret/launchcodes.txt'
[root@cXn5 ~]# getfacl /mnt/firstvol/supersecret/launchcodes.txt
getfacl: Removing leading '/' from absolute path names
# file: mnt/firstvol/supersecret/launchcodes.txt
# owner: acluser
# group: acluser
user::rw-
user:acluser:rw-   #effective:rw-
group:---
mask::rw-
other:---
```



Performance Checklist

Working with Quotas

Lab Overview: In this exercise you will enable, set, and enforce quotas on your **replvol** volume.

Success Criteria: You will have successfully completed this lab when you enabled quotas for your **replvol** volume with a 256M quota set for **replvol/quotadir**.

Before you begin...

Make sure the **replvol** volume is active and mounted on **cXn5**.

- ❑ 1. Enable quotas for your **replvol** volume.

```
[root@cXn1 ~]# gluster volume quota replvol enable
```

```
Enabling quota has been successful
```

- ❑ 2. Set a 256 MiB limit for the **/quotadir** directory on your **replvol** volume.

```
[root@cXn1 ~]# gluster volume quota replvol limit-usage /quotadir 256MB
limit set on /quotadir
```

- ❑ 3. From your **cXn5** machine, create the **/mnt/replvol/quotadir** directory then create a 128MiB file called **shiny** in that directory.

```
[root@cXn5 ~]# mkdir /mnt/replvol/quotadir
[root@cXn5 ~]# dd if=/dev/zero of=/mnt/replvol/quotadir/shiny bs=1M count=128
```

- ❑ 4. On **cXn1** display a quota overview for your **replvol** volume.

```
[root@cXn1 ~]# gluster volume quota replvol list
```

path	limit_set	size
/quotadir	256MB	128.0MB

- ❑ 5. On your **cXn5** machine attempt to create a 512 MiB file called **/mnt/replvol/quotadir/toolarge**. This should fail after approximately 128 MiB.

```
[root@cXn5 ~]# dd if=/dev/zero of=/mnt/replvol/quotadir/toolarge bs=1M count=512
dd: writing '/mnt/replvol/quotadir/toolarge': Input/output error
149+0 records in
148+0 records out
155189248 bytes (155 MB) copied, 14.324 s, 10.8 MB/s
```



Case Study

Quotas and ACLs

Lab Overview: In this exercise you will create a shared directory with both quotas and ACLs on your **firstvol** volume.

Success Criteria: You will have successfully completed this lab when you have created a shared directory on your **firstvol** volume using the criteria below.

Before you begin...

Make sure you still have a working **firstvol** volume mounted on **/mnt/firstvol** on your **cXn5** machine.

You have been asked to create a shared directory for the (new) **architects** group. This new directory should be created with the following criteria:

- Location: **firstvol/architects**
- Ownership: **root:architects**
- Permissions:

- All files and directories in **architects** should be owned by the group **architects**.
- People in the **architects** group should have full access to all files and directories in **firstvol/architects**, including any newly created files and directories.
- People in the (new) **managers** group should have read-only access to **firstvol/architects** and any new files and/or directories created there. (This includes *execute* permission for directories.)
- People outside of the **architects** and **managers** groups should have no access to the **firstvol/architects** directory or any files and directories located below that directory. (It is understood that **root** has full access.)
- Any new files or directories created under the **firstvol/architects** directory should automatically belong to the **architects** group. You will probably want to use **setgid** permission to fulfill this requirement.
- Disk usage for the **firstvol/architects** directory should be limited to a maximum of 512 MiB.

As part of these criteria you will have to create both the **architects** and **managers** groups on your **cXn5** machine.

1. First create the necessary groups on **cXn5**:

```
[root@cXn5 ~]# groupadd architects
[root@cXn5 ~]# groupadd managers
```

2. If you successfully completed the ACL exercise earlier, your **firstvol** volume should already be mounted with ACL support. If not, add the **acl** mount option to the **/etc/fstab** entry for your **firstvol** volume on **cXn5**, then unmount and mount your **firstvol** volume.

Make sure your **/etc/fstab** entry looks something like this:

```
cXn1.example.com:/firstvol /mnt/firstvol    glusterfs _netdev,acl 0 0
```

Then unmount and mount your **firstvol** volume.

```
[root@cXn5 ~]# umount /mnt/firstvol
[root@cXn5 ~]# mount /mnt/firstvol
```

3. On your **cXn5** machine create the **/mnt/firstvol/architects** directory with **2770** permissions (**setgid**, **rwX** for user, **rwX** for group, nothing for other).

```
[root@cXn5 ~]# mkdir -m2770 /mnt/firstvol/architects
```

4. Change the group ownership of your new directory to **architects**.

```
[root@cXn5 ~]# chown :architects /mnt/firstvol/architects
```

5. Set the full-access ACL and default ACL for the **architects** group.

```
[root@cXn5 ~]# setfacl -m g:architects:rwX /mnt/firstvol/architects
[root@cXn5 ~]# setfacl -m d:g:architects:rwX /mnt/firstvol/architects
```

6. Set the read-only ACL and default ACL for the **managers** group. Do not forget the **execute** permissions.

```
[root@cXn5 ~]# setfacl -m g:managers:rx /mnt/firstvol/architects
[root@cXn5 ~]# setfacl -m d:g:managers:rx /mnt/firstvol/architects
```

7. From your **cXn1** machine, enable quotas for your **firstvol** volume.

```
[root@cXn1 ~]# gluster volume quota firstvol enable
```

8. Set a directory quota of 512 MiB for the **/architects** directory on your **firstvol** volume.

```
[root@cXn1 ~]# gluster volume quota firstvol limit-usage /architects 512MB
```

Extending Volumes



Performance Checklist

Extending a Volume

Lab Overview: In this exercise you will extend your **firstvol** volume to use two extra bricks.

Success Criteria: You will have successfully completed this lab when your **firstvol** has a total of four bricks.

Before you begin...

Make sure you still have a working **firstvol** volume.

- 1. Begin by creating two 2 GiB bricks, **/dev/vg_bricks/brick9** on **cXn1**, mounted on **/bricks/brick9**, and **/dev/vg_bricks/brick10** on **cXn2**, mounted on **/bricks/brick10**.

First create the two logical volumes:

```
[root@cXn1 ~]# lvcreate -L 2G -n brick9 vg_bricks
```

```
[root@cXn2 ~]# lvcreate -L 2G -n brick10 vg_bricks
```

Next create XFS file systems on the new logical volumes, making sure to use 512 byte inodes.

```
[root@cXn1 ~]# mkfs.xfs -i size=512 /dev/vg_bricks/brick9
meta-data=/dev/vg_bricks/brick9 isize=512    agcount=4, agsize=131072 blks
         =                       sectsz=512    attr=2
data      =                       bsize=4096   blocks=524288, imaxpct=25
         =                       sunit=0      swidth=0 blks
naming    =version 2              bsize=4096   ascii-ci=0
log       =internal log          bsize=4096   blocks=2560, version=2
         =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime  =none                  extsz=4096   blocks=0, rtextents=0
```

```
[root@cXn2 ~]# mkfs.xfs -i size=512 /dev/vg_bricks/brick10
meta-data=/dev/vg_bricks/brick10 isize=512    agcount=4, agsize=131072 blks
         =                       sectsz=512    attr=2
data      =                       bsize=4096   blocks=524288, imaxpct=25
         =                       sunit=0      swidth=0 blks
naming    =version 2              bsize=4096   ascii-ci=0
log       =internal log          bsize=4096   blocks=2560, version=2
         =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime  =none                  extsz=4096   blocks=0, rtextents=0
```

Create the mount points:

```
[root@cXn1 ~]# mkdir /bricks/brick9
```

```
[root@cXn2 ~]# mkdir /bricks/brick10
```

Add the following lines to **/etc/fstab**:

cXn1

```
/dev/vg_bricks/brick9 /bricks/brick9 xfs defaults 0 2
```

cXn2

```
/dev/vg_bricks/brick10 /bricks/brick10 xfs defaults 0 2
```

On both of the nodes mount your new file system:

```
[root@cXn1 ~]# mount -a
```

```
[root@cXn2 ~]# mount -a
```

- 2. Add your two new bricks to your **firstvol** volume. Make sure to keep the volume type as *Distributed*.

```
[root@cXn1 ~]# gluster volume add-brick firstvol \  
> cXn1.example.com:/bricks/brick9 \  
> cXn2.example.com:/bricks/brick10  
Add Brick successful
```

- 3. Initiate a *rebalance* of your **firstvol** volume.

```
[root@cXn1 ~]# gluster volume rebalance firstvol start  
Starting rebalance on volume firstvol has been successful
```

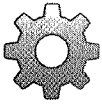
- 4. Wait for the rebalance to complete.

```
[root@cXn1 ~]# gluster volume rebalance firstvol status
```

Node	Rebalanced-files	size	scanned	failures	status
localhost	1	0	12	0	completed
cXn2.example.com	3	0	14	0	completed
cXn4.example.com	0	0	11	0	completed
cXn3.example.com	0	0	11	0	completed

- 5. Inspect the contents of your different bricks, what do you notice?

Some files have been moved to the new bricks but top-level directories are present on all bricks.



Case Study

Extending a Volume

Lab Overview: In this exercise you will grow your **replvol** volume into a 2x2 Distributed-Replicate volume.

Success Criteria: You will have successfully completed this lab when your **replvol** is a 2x2 Distributed-Replicate volume.

Before you begin...

Make sure you still have a working **replvol** volume.

You have been asked to grow your **replvol** volume into a 2x2 Distributed-Replicate volume.

In order to accomplish this task you should use two new 2 GiB bricks. On **cXn3** you should add a new brick on **/dev/vg_bricks/brick11**, mounted on **/brick/brick11**, and on **cXn4** you should add a new brick on **/dev/vg_brick/brick12**, mounted on **/bricks/brick12**.

Any new files written to your brick should be distributed according to the new layout, and any existing files should be re-distributed.

1. Create the new bricks. Execute the following commands on both your **cXn3** and **cXn4** machines, replacing **Y** with the node number (3 or 4), and **Y+8** with the the node number plus 8 (11 or 12).

```
[root@cXnY ~]# lvcreate -L 2G -n brickY+8 vg_bricks
Logical volume "brickY+8" created
[root@cXnY ~]# mkfs.xfs -i size=512 /dev/vg_bricks/brickY+8
meta-data=/dev/vg_bricks/brickY+8 isize=512    agcount=4, agsize=131072 blks
       =                               sectsz=512   attr=2
data     =                               bsize=4096   blocks=524288, imaxpct=25
       =                               sunit=0      swidth=0 blks
naming   =version 2                     bsize=4096   ascii-ci=0
log      =internal log                  bsize=4096   blocks=2560, version=2
       =                               sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                          extsz=4096   blocks=0, rtextents=0
[root@cXnY ~]# mkdir /bricks/brickY+8
[root@cXnY ~]# echo "/dev/vg_bricks/brickY+8 /bricks/brickY+8 xfs defaults 0 2" >> /
etc/fstab
[root@cXnY ~]# mount -a
```

2. Add your two new bricks to your volume, while keeping the *replicate* count set to **2**:

```
[root@cXn1 ~]# gluster volume add-brick replvol \
> cXn3.example.com:/bricks/brick11 \
> cXn4.example.com:/bricks/brick12
Add Brick successful
```

3. Re-balance the volume to distribute data and to make sure that new files will be written to their correct bricks.

```
[root@cXn1 ~]# gluster volume rebalance replvol start
Starting rebalance on volume replvol has been successful
```

4. Wait for the rebalance operation to complete.

[root@cXn1 ~]# gluster volume rebalance replvol status					
Node	Rebalanced-files	size	scanned	failures	status
localhost	0	0	12	0	completed
cXn4.example.com	0	0	13	0	completed
cXn3.example.com	7	268562432	19	0	completed
cXn2.example.com	0	0	12	0	completed

IP Failover



Performance Checklist

Configuring IP Failover

Lab Overview: In this exercise you will configure IP failover for your NFS and CIFS exports.

Success Criteria: You will have successfully completed this exercise when **CTDB** is fully configured and running.

Before you begin...

Make sure all of your nodes are still operational.

- 1. First we will need a replicated volume to store our metadata for **CTDB**. On your **cXn1** and **cXn2** machines create a new 256 MiB logical volume called **brickY+12**, where **Y+12** is the node number plus 12, format them with an XFS file system, and mount them on **/bricks/brick13** and **/bricks/brick14** respectively.

Run the following commands on both of your **cXn1** and **cXn2** machines. Make sure to replace **Y** with the correct node number and **Y+12** with the node number with 12 added to it.

```
[root@cXnY ~]# lvcreate -L 256M -n brickY+12 vg_bricks
[root@cXnY ~]# mkfs.xfs -i size=512 /dev/vg_bricks/brickY+12
meta-data=/dev/vg_bricks/brickY+12 isize=512    agcount=4, agsize=16384 blks
       =                               sectsz=512   attr=2
data     =                               bsize=4096   blocks=65536, imaxpct=25
       =                               sunit=0      swidth=0 blks
naming   =version 2                       bsize=4096   ascii-ci=0
log      =internal log                   bsize=4096   blocks=1200, version=2
       =                               sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                           extsz=4096   blocks=0, rtextents=0
[root@cXnY ~]# mkdir /bricks/brickY+12
[root@cXnY ~]# echo "/dev/vg_bricks/brickY+12 /bricks/brickY+12 xfs defaults 0 2"
>> /etc/fstab
[root@cXnY ~]# mount -a
```

- 2. Combine your two new bricks into a two-way replicated volume called **ctdbmeta**. Do not start this volume right now.

```
[root@cXn1 ~]# gluster volume create ctdbmeta replica 2 \
> cXn1.example.com:/bricks/brick13 \
> cXn2.example.com:/bricks/brick14
Creation of volume ctdbmeta has been successful. Please start the volume to access
data.
```

- 3. Verify your new volume using **gluster volume info**.

```
[root@cXn1 ~]# gluster volume info ctdbmeta

Volume Name: ctdbmeta
Type: Replicate
Volume ID: 9125146f-31d9-4ff0-96ad-cb386b7c48e6
Status: Created
```

```
Number of Bricks: 1 x 2 = 2
Transport-type: tcp
Bricks:
Brick1: cXn1.example.com:/bricks/brick13
Brick2: cXn2.example.com:/bricks/brick14
```

- 4. On all four of your nodes, (**cXn1** through **cXn4**), change **META="all"** to **META="ctdbmeta"** in both **/var/lib/glusterd/hooks/1/start/post/S29CTDBsetup.sh** and **/var/lib/glusterd/hooks/1/stop/pre/S29CTDB-teardown.sh**.

Run these two commands on all four of your nodes:

```
[root@cXnY ~]# sed -i 's/^META="all"/META="ctdbmeta"/' /var/lib/glusterd/hooks/1/
start/post/S29CTDBsetup.sh
[root@cXnY ~]# sed -i 's/^META="all"/META="ctdbmeta"/' /var/lib/glusterd/hooks/1/
stop/pre/S29CTDB-teardown.sh
```

- 5. Start your **ctdbmeta** volume.

```
[root@cXn1 ~]# gluster volume start ctdbmeta
Starting volume ctdbmeta has been successful
```

- 6. Your new volume should now be mounted on **/gluster/lock**. Verify that this is the case on all four of your nodes.

```
[root@cXnY ~]# df -h /gluster/lock
Filesystem      Size  Used Avail Use% Mounted on
cXnY.example.com:ctdbmeta
                252M   13M  239M   6% /gluster/lock
```

- 7. Create the **/gluster/lock/ctdb** file with the following contents:

```
CTDB_RECOVERY_LOCK=/gluster/lock/lockfile
CTDB_PUBLIC_ADDRESSES=/etc/ctdb/public_addresses
CTDB_MANAGES_SAMBA=yes
CTDB_NODES=/etc/ctdb/nodes
```

```
[root@cXn1 ~]# cat > /gluster/lock/ctdb << EOF
> CTDB_RECOVERY_LOCK=/gluster/lock/lockfile
> CTDB_PUBLIC_ADDRESSES=/etc/ctdb/public_addresses
> CTDB_MANAGES_SAMBA=yes
> CTDB_NODES=/etc/ctdb/nodes
> EOF
```

- 8. Create a new **/gluster/lock/nodes** file and list the IP addresses of your four nodes in it, one per line.

```
[root@cXn1 ~]# for IP in 192.168.0.X{1..4}; do echo ${IP}; done >> /gluster/lock/
nodes;
```


- 9. Create a new file called **/gluster/lock/public_addresses** with the following content:

```
192.168.0.X6/24 eth0
```

This means that we want to use the floating IP address **192.168.0.X6**, with a prefix (subnetmask) of **24** and it should be assigned to the **eth0** interface when in use.

```
[root@cXn1 ~]# echo "192.168.0.X6/24 eth0" > /gluster/lock/public_addresses
```

- 10. On all four your nodes create a symlink from **/etc/sysconfig/ctdb** to **/gluster/lock/ctdb**, from **/etc/ctdb/nodes** to **/gluster/lock/nodes**, and from **/etc/ctdb/public_addresses** to **/gluster/lock/public_addresses**. You might have to remove an existing **/etc/sysconfig/ctdb** file first.

```
[root@cXnY ~]# rm /etc/sysconfig/ctdb
[root@cXnY ~]# ln -s /gluster/lock/ctdb /etc/sysconfig/ctdb
[root@cXnY ~]# ln -s /gluster/lock/nodes /etc/ctdb/nodes
[root@cXnY ~]# ln -s /gluster/lock/public_addresses /etc/ctdb/public_addresses
```

- 11. Start the **ctdb** service on all four of your nodes.

```
[root@cXnY ~]# service ctdb start
```

- 12. Now you can test if your failover works. On your **cXn5** machine, unmount **/mnt/replvol** then change the **/etc/fstab** entry for **replvol** to use the floating IP address you configured (**192.168.0.X6**).

```
[root@cXn5 ~]# umount /mnt/replvol
```

Now change the entry in **/etc/fstab** to the following:

```
192.168.0.X6:/replvol /mnt/replvol nfs vers=3 0 0
```

- 13. Mount the **replvol** volume on **cXn5**, then verify that you can access the volume.

```
[root@cXn5 ~]# mount /mnt/replvol
[root@cXn5 ~]# ls -l /mnt/replvol
```

- 14. Determine which of your hosts currently has the floating IP address. To do this use the **ip a s eth0** command on all four of your nodes to determine which host has the floating IP address. Another options is to use the **ctdb ip** command.

```
[root@cXn1 ~]# ctdb ip
Public IPs on node 0
192.168.0.X6 node[0] active[eth0] available[eth0] configured[eth0]
```

- 15. Shutdown the node that currently has the public IP (in our example this is the **cXn1** machine, but you might have to shutdown a different machine).

```
[root@cXn1 ~]# poweroff
```

- 16. Attempt to access `/mnt/replvol` from your **cXn5** machine. This operation should stall for a number of seconds, then continue.

```
[root@cXn5 ~]# ls -l /mnt/replvol
```

- 17. **Important Cleanup:** Restart the node that you shut down during this lab.

In a physical classroom open a terminal on your desktopX machine, then execute the following commands (where **Z** is the number of the node you shut down earlier:

```
[student@desktopX ~]$ su -  
Password: redhat  
[root@desktopX ~]# virsh start nodeZ
```

In a virtual classroom, open the console for the machine you shut down earlier, then click on the **Start** button.

Geo-Replication



Performance Checklist

Configure Geo-Replication

Lab Overview: In this exercise you will configure Geo-Replication between your **firstvol** volume on **cXn1** and a slave volume, **slavevol**, on **cXn5**

Success Criteria: You will have successfully completed this lab when geo-replication is working for your **firstvol** volume.

Before you begin...

Make sure your **firstvol** volume is still working.

- ❑ 1. Create an SSH key that will be used for geo-replication. Be sure you store it in **/var/lib/glusterd/geo-replication/secret.pem** on **cXn1**.

```
[root@cxn1 ~]# ssh-keygen -f /var/lib/glusterd/geo-replication/secret.pem
Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase): Enter
Enter same passphrase again: Enter
Your identification has been saved in /var/lib/glusterd/geo-replication/secret.pem.
Your public key has been saved in /var/lib/glusterd/geo-replication/secret.pem.pub.
The key fingerprint is:
c3:72:e7:d0:ba:79:e8:34:6d:ef:c1:ce:f7:6c:56:b0 root@cxn1.example.com
The key's randomart image is:
+---[ RSA 2048]-----+
|                                     |
|                                     |
|                                     |
|      . .   .                      |
|    , S o   o |                    |
|    o B . E . |                    |
|      +,+ o   . |                   |
|     ..=,+ ...o |                   |
|    .+. .=. +o |                   |
+-----+-----+
```

- ❑ 2. Edit `/var/lib/glusterd/geo-replication/secret.pem.pub` so that only the `/usr/libexec/glusterfs/gsync` command can be executed when using this key.

Edit the `/var/lib/glusterd/geo-replication/secret.pem.pub` file so that it starts with the following. Make sure not to change anything after `ssh-rsa`.

```
command="/usr/libexec/glusterfs/gsyncd" ssh-rsa AAAAB3NzaC1...
```

- ❑ 3. Create a new group on **cXn5** called **geogroup** and a new user called **geoaccount** that is a member of the group **geogroup**. Set the password for **geoaccount** to **redhat**.

```
[root@cXn5 ~]# groupadd geogroup
[root@cXn5 ~]# useradd -G geogroup geoaccount
[root@cXn5 ~]# echo redhat | passwd --stdin geoaccount
```

- ❑ 4. From your **cXn1** machine, copy the SSH publickey you just created to the **geoaccount** account on **cXn5**.

```
[root@cXn1 ~]# ssh-copy-id -i /var/lib/glusterd/geo-replication/secret.pem.pub
geoaccount@cXn5.example.com
```

- ❑ 5. On **cXn5** create a new 8 GiB logical volume called **slavebrick1** in the **vg_bricks** volume group, format it with an XFS file system with 512 byte inodes, and persistently mount it on **/bricks/slavebrick1**

```
[root@cXn5 ~]# lvcreate -L 8G -n slavebrick1 vg_bricks
[root@cXn5 ~]# mkfs.xfs -i size=512 /dev/vg_bricks/slavebrick1
[root@cXn5 ~]# mkdir -p /bricks/slavebrick1
[root@cXn5 ~]# echo "/dev/vg_bricks/slavebrick1 /bricks/slavebrick1 xfs defaults 0
2" >> /etc/fstab
[root@cXn5 ~]# mount -a
```

- ❑ 6. On **cXn5** create a new volume called **slavevol** using the brick you just created, then start it.

```
[root@cXn5 ~]# gluster volume create slavevol cXn5.example.com:/bricks/slavebrick1
[root@cXn5 ~]# gluster volume start slavevol
```

- ❑ 7. On **cXn5** create a new directory, **/var/mountbroker-root**, to act as a root mount point for the mount broker.

```
[root@cXn5 ~]# mkdir -p /var/mountbroker-root
```

- ❑ 8. On **cXn5** add the following lines to the **volume management** section of **/etc/glusterfs/glusterd.vol**:

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

The lines allow **geoaccount** access to the **slavevol** volume using the mount broker and allow members of the group **geogroup** to read the log files.

When you are done with your changes your **/etc/glusterfs/glusterd.vol** file should look something like this:

```
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
```

- 9. Restart the **glusterd** daemon on **cXn5** to activate your changes.

```
[root@cXn5 ~]# service glusterd restart
```

- 10. From **cXn1** initiate geo-replication between the **firstvol** volume and the **slavevol** volume on **cXn5** using the **geoaccount** account and the mount broker.

```
[root@cXn1 ~]# gluster volume geo-replication firstvol
geoaccount@cXn5.example.com::slavevol start
Starting geo-replication session between firstvol &
geoaccount@cXn5.example.com::slavevol has been successful
```

- 11. Inspect the status of your replication.

```
[root@cXn1 ~]# gluster volume geo-replication firstvol
geoaccount@cXn5.example.com::slavevol status
```

MASTER	SLAVE	STATUS
firstvol	geoaccount@cXn5.example.com::slavevol	Starting

- 12. Wait for the initial replication to finish.

```
[root@cXn1 ~]# gluster volume geo-replication firstvol
geoaccount@cXn5.example.com::slavevol status
```

MASTER	SLAVE	STATUS
firstvol	geoaccount@cXn5.example.com::slavevol	OK



Note

Due to a bug it can happen that your Geo-Replication reports a state of **faulty**, even when you have configured everything as it should be. If this happen to you, check if all your files are synchronized, including any new files you might create. If files do get replicated, but the status stays at **faulty** just pretend that the status reads **OK** for this lab.

Unified File and Object Storage



Performance Checklist

Configuring Swift

Lab Overview: In this exercise you will configure Swift for unified file and object storage.

Success Criteria: You will have successfully completed this lab when you have a working Swift cluster.

Before you begin...

Make sure you still have a working cluster with a working **firstvol** volume.

- ❑ 1. On your **cXn1** machine add an admin user for your **firstvol** volume to the file **/etc/swift/proxy-server.conf** called **ufouser**, with a password of **redhat**.

Add the following line to the **[filter:tempauth]** section:

```
user_firstvol_ufouser = redhat .admin
```

- ❑ 2. On your **cXn1** machine configure **/etc/swift/proxy-server.conf** to use **memcached** servers on all four of your nodes. Use the default port for **memcached**.

Add the following line to the **[filter:cache]** section:

```
memcache_servers = 192.168.0.X1:11211, 192.168.0.X2:11211, 192.168.0.X3:11211,
192.168.0.X4:11211
```

- ❑ 3. From your **cXn1** machine, copy the **/etc/swift/proxy-server.conf** file to your other three nodes.

```
[root@cXn1 ~]# for NODE in cXn[2..4]; do \
> scp /etc/swift/proxy-server.conf ${NODE}:/etc/swift/
> done
```

- ❑ 4. On all four of your nodes enable and start the **memcached** service.

Run the following commands on all four of your nodes:

```
[root@cXnY ~]# chkconfig memcached on
[root@cXnY ~]# service memcached start
```

- ❑ 5. On all four of your nodes enable the **gluster-swift-account**, **gluster-swift-container**, **gluster-swift-object**, and **gluster-swift-proxy** services.

Run the following commands on all four of your nodes:

```
[root@cXnY ~]# for SERVICE in gluster-swift-{account,container,object,proxy}; do
> chkconfig ${SERVICE} on
```

```
> done
```

- ❑ 6. On all four of your nodes start all of the **gluster-swift-*** services.

Run the following commands on all four of your nodes:

```
[root@cXnY ~]# swift-init main start
```



Performance Checklist

Testing Swift

Lab Overview: In this exercise you will use the **curl** command to test the Swift REST-API.

Success Criteria: You will have successfully completed this lab when you have successfully retrieved and stored files from your **firstvol** volume using the Swift REST API.

Before you begin...

Make sure you have successfully configured Swift and your **firstvol** volume is still operational.

Perform all of the following tasks on your **cXn5** machine.

- ❑ 1. Use **curl** to obtain an authentication token for your **ufouser** user on your **firstvol** volume. Store the entire **X-Auth-Token** header in a shell variable called **MYAUTH**.

```
[root@cXn5 ~]# curl -v -H 'X-Storage-User: firstvol:ufouser' -H 'X-Storage-
Pass:redhat' http://cXn1.example.com:8080/auth/v1.0
* About to connect() to cXn1.example.com port 8080 (#0)
*   Trying 192.168.0.X1... connected
* Connected to cXn1.example.com (192.168.0.X1) port 8080 (#0)
> GET /auth/v1.0 HTTP/1.1
> User-Agent: curl/7.19.7 (x86_64-redhat-linux-gnu) libcurl/7.19.7 NSS/3.X.1.0
zlib/1.2.3 libidn/1.18 libssh2/1.2.2
> Host: cXn1.example.com:8080
> Accept: */*
> X-Storage-User: firstvol:ufouser
> X-Storage-Pass:redhat
>
< HTTP/1.1 200 OK
< X-Storage-Url: http://127.0.0.1:8080/v1/AUTH_firstvol
< X-Storage-Token: AUTH_tk4528ddb35fa7493d9d898acb163e679d
< X-Auth-Token: AUTH_tk4528ddb35fa7493d9d898acb163e679d
< Content-Length: 0
< Date: Tue, 10 Nov 2012 10:10:32 GMT
<
* Connection #0 to host cXn1.example.com left intact
* Closing connection #0
[root@cXn5 ~]# MYAUTH="X-Auth-Token: AUTH_tk4528ddb35fa7493d9d898acb163e679d"
```

Or if you want to script this:

```
[root@cXn5 ~]# MYAUTH=$(curl -v -H 'X-Storage-User: firstvol:ufouser' -H 'X-
Storage-Pass:redhat' http://cXn1.example.com:8080/auth/v1.0 2>&1 | awk '/X-Auth-
Token/{print $2,$3}')
```

- 2. Use **curl** with the authentication token you just retrieved to request a list of containers on your **firstvol** volume. Remember you should refer to that volume as **AUTH_firstvol** when using Swift.

```
[root@cXn5 ~]# curl -X GET -H "${MYAUTH}" http://cXn1.example.com:8080/v1/AUTH_firstvol/
architects
supersecret
```

- 3. Use **curl** to request the contents of the **supersecret/launchcodes.txt** file on your **firstvol** volume.

```
[root@cXn5 ~]# curl -X GET -H "${MYAUTH}" http://cXn1.example.com:8080/v1/AUTH_firstvol/supersecret/launchcodes.txt
00000000
```

- 4. Use **curl** to create a new file on your **firstvol** volume called **/supersecret/passwd**. Upload the contents of your **/etc/passwd** file on **cXn5** into that new file.

```
[root@cXn5 ~]# curl -X PUT -H "${MYAUTH}" http://cXn1.example.com:8080/v1/AUTH_firstvol/supersecret/passwd --data-binary @/etc/passwd
<html>
<head>
  <title>201 Created</title>
</head>
<body>
  <h1>201 Created</h1>
  <br /><br />

</body>
</html>
```

- 5. Grant the *imaginary* users **alice**, **bob**, and **mary** write access to the **/supersecret** directory on **firstvol**.

```
[root@cXn5 ~]# curl -X POST -H "${MYAUTH}" http://cXn1.example.com:8080/v1/AUTH_firstvol/supersecret -H 'X-Container-Write: alice,bob,mary'
```


Troubleshooting



Performance Checklist

Investigate Self-Heal

Lab Overview: In this exercise you will investigate the *Self-Heal* feature of Red Hat Storage Server.

Before you begin...

Make sure you still have a working **replvol** volume mounted on **/mnt/replvol** on your **cXn5** machine using a floating IP address.

- ❑ 1. Use your **cXn3** machine to investigate which bricks make up your **replvol** volume.

```
[root@cXn3 ~]# gluster volume info replvol

Volume Name: replvol
Type: Distributed-Replicate
Volume ID: 1f32b6e6-a409-4d5f-aa88-793c4a770253
Status: Started
Number of Bricks: 2 x 2 = 4
Transport-type: tcp
Bricks:
Brick1: cXn3.example.com:/bricks/brick3
Brick2: cXn4.example.com:/bricks/brick4
Brick3: cXn3.example.com:/bricks/brick11
Brick4: cXn4.example.com:/bricks/brick12
Options Reconfigured:
auth.allow: 192.168.0.*
auth.reject: 192.168.0.X0
features.quota: on
features.limit-usage: /quotadir:256MB
```

If everything is as expected you should see **brick3** and **brick11** on **cXn3** and **brick4** and **brick12** on **cXn4**.

In this configuration **brick3** and **brick11** are replicated to each other and files sent to the volume are distributed between this mirror and the mirror on **brick4** and **brick12**.

- ❑ 2. Power down your **cXn3** machine.

```
[root@cXn3 ~]# poweroff
```

- ❑ 3. On your **cXn5** machine create 10 new 1 MiB files called **/mnt/replvol/testfile1** through **/mnt/replvol/testfile10**.

```
[root@cXn5 ~]# for FILE in /mnt/replvol/testfile{1..10}; do
> dd if=/dev/zero of=${FILE} bs=1M count=1
> done
```

- ❑ 4. On your **cXn4** machine inspect the contents of **/bricks/brick4** and **/bricks/brick12**. You should see the various **testfile*** files distributed (somewhat) evenly between the two.

```
[root@cXn4 ~]# ls -l /bricks/brick4
[root@cXn4 ~]# ls -l /bricks/brick12
```

- 5. Power on your **cXn3** machine.

In a physical classroom you can do this by opening a terminal on your **desktopX** machine as **root** and executing the command **virsh start node3**.

```
[root@desktopX ~]# virsh start node3
```

In a virtual learning environment you can open the console for your **cXn3** (node3) machine and click on the **Start** button above the console.

- 6. Once **cXn3** is back on-line, log in as **root** and inspect the contents of **/bricks/brick3** and **/bricks/brick11**. What do you notice?

```
[root@cXn3 ~]# ls -l /bricks/brick3
[root@cXn3 ~]# ls -l /bricks/brick11
```

Even though the new files have been created on the bricks, their size is still 0 bytes.

- 7. Request a list of files on your **replvol** volume that still need to be healed.

```
[root@cXn3 ~]# gluster volume heal replvol info
Heal operation on volume replvol has been successful

Brick cXn3.example.com:/bricks/brick3
Number of entries: 0

Brick cXn4.example.com:/bricks/brick4
Number of entries: 3
/testfile5
/testfile7
/testfile9

Brick cXn3.example.com:/bricks/brick11
Number of entries: 0

Brick cXn4.example.com:/bricks/brick12
Number of entries: 7
/testfile1
/testfile2
/testfile3
/testfile4
/testfile6
/testfile8
/testfile10
```

- 8. Wait for the self-heal daemon to run, or trigger a self heal for your **replvol** volume if you are impatient.

```
[root@cXn3 ~]# gluster volume heal replvol
Heal operation on volume replvol has been successful
```

- 9. Request a list of files that have been healed on the **replvol** volume.

```
[root@cXn3 ~]# gluster volume heal replvol info healed
Heal operation on volume replvol has been successful

Brick cXn3.example.com:/bricks/brick3
Number of entries: 0

Brick cXn4.example.com:/bricks/brick4
Number of entries: 3
at          path on brick
-----
2012-11-13 12:12:13 /testfile9
2012-11-13 12:12:13 /testfile7
2012-11-13 12:12:13 /testfile5

Brick cXn3.example.com:/bricks/brick11
Number of entries: 0

Brick cXn4.example.com:/bricks/brick12
Number of entries: 7
at          path on brick
-----
2012-11-13 12:12:13 /testfile10
2012-11-13 12:12:13 /testfile8
2012-11-13 12:12:13 /testfile6
2012-11-13 12:12:13 /testfile4
2012-11-13 12:12:13 /testfile3
2012-11-13 12:12:13 /testfile2
2012-11-13 12:12:13 /testfile1
```

- 10. Check if any files failed the self-heal.

```
[root@cXn3 ~]# gluster volume heal replvol info heal-failed
Heal operation on volume replvol has been successful

Brick cXn3.example.com:/bricks/brick3
Number of entries: 0

Brick cXn4.example.com:/bricks/brick4
Number of entries: 0

Brick cXn3.example.com:/bricks/brick11
Number of entries: 0

Brick cXn4.example.com:/bricks/brick12
Number of entries: 0
```

- 11. Check the contents of **/bricks/brick3** and **/bricks/brick11** on your **cXn3** machine. Do the files now have the correct size?

```
[root@cXn3 ~]# ls -l /bricks/brick3
[root@cXn3 ~]# ls -l /bricks/brick11
```

Yes, all files are now at their correct size.



Case Study

Replacing a Brick

Lab Overview: In this exercise you will replace a brick in your **replvol** volume without causing any downtime.

Success Criteria: You will have successfully completed this lab when you have successfully replaced a brick in your **replvol** volume using the criteria below.

Before you begin...

Make sure you still have a working **replvol** volume.

The storage backing **/bricks/brick3** on your **cXn3** machine has been acting up lately. You have been asked to replace that brick with a new 2 GiB brick mounted on **/bricks/brick17** on your **cXn1** machine without causing any downtime or service disruption. It should be backed by a logical volume called **/dev/vg_bricks/brick17**.

After the replacement is complete, remove the old logical volume **/dev/vg_bricks/brick3** on your **cXn3** machine.

1. Begin by creating the new brick on your **cXn1** machine.

```
[root@cXn1 ~]# lvcreate -L 2G -n brick17 vg_bricks
Logical volume "brick17" created
[root@cXn1 ~]# mkfs.xfs -i size=512 /dev/vg_bricks/brick17
meta-data=/dev/vg_bricks/brick17 isize=512    agcount=4, agsize=131072 blks
       =                               sectsz=512   attr=2
       =                               bsize=4096   blocks=524288, imaxpct=25
       =                               sunit=0     swidth=0 blks
naming   =version 2                bsize=4096   ascii-ci=0
log      =internal log             bsize=4096   blocks=2560, version=2
       =                               sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                     extsz=4096   blocks=0, rtextents=0
[root@cXn1 ~]# mkdir /bricks/brick17
[root@cXn1 ~]# echo "/dev/vg_bricks/brick17 /bricks/brick17 xfs defaults 0 2" >> /
etc/fstab
[root@cXn1 ~]# mount -a
```

2. Initiate the replacement operation.

```
[root@cXn1 ~]# gluster volume replace-brick replvol \
> cxn3.example.com:/bricks/brick3 \
> cxn1.example.com:/bricks/brick17 \
> start
replace-brick started successfully
```

3. Wait for the migration to complete.

```
[root@cXn1 ~]# gluster volume replace-brick replvol \
> cxn3.example.com:/bricks/brick3 \
> cxn1.example.com:/bricks/brick17 \
> status
Number of files migrated = 9           Migration complete
```

4. Commit your changes.

```
[root@cXn1 ~]# gluster volume replace-brick replvol \  
> cxn3.example.com:/bricks/brick3 \  
> cxn1.example.com:/bricks/brick17 \  
> commit  
replace-brick commit successful
```

5. Unmount the old brick.

```
[root@cXn3 ~]# umount /bricks/brick3
```

6. Remove the entry for the old brick from **/etc/fstab** on your **cXn3** machine.

```
[root@cXn3 ~]# sed -i '/brick3/d' /etc/fstab
```

7. Remove the **/bricks/brick3** mount point from your **cXn3** machine.

```
[root@cXn3 ~]# rmdir /bricks/brick3
```

8. Remove the **/dev/vg_bricks/brick3** logical volume from your **cXn3** machine.

```
[root@cXn3 ~]# lvremove /dev/vg_bricks/brick3  
Do you really want to remove active logical volume brick3? [y/n]: y  
Logical volume "brick3" successfully removed
```



Personal Notes