



Red Hat Reference Architecture Series

Deploying a Highly Available OpenShift Enterprise 1.2 Environment

Using Red Hat Enterprise Virtualization 3.2 and Red
Hat Storage Server 2.1

John Herr, Senior Software Engineer
RHCA, RHCVA

Version 1.0

November 2013





1801 Varsity Drive™
Raleigh NC 27606-2072 USA
Phone: +1 919 754 3700
Phone: 888 733 4281
Fax: +1 919 754 3701
PO Box 13588
Research Triangle Park NC 27709 USA

Linux is a registered trademark of Linus Torvalds. Red Hat, Red Hat Enterprise Linux and the Red Hat "Shadowman" logo are registered trademarks of Red Hat, Inc. in the United States and other countries.

UNIX is a registered trademark of The Open Group.

Intel, the Intel logo and Xeon are registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

All other trademarks referenced herein are the property of their respective owners.

© 2013 by Red Hat, Inc. This material may be distributed only subject to the terms and conditions set forth in the Open Publication License, V1.0 or later (the latest version is presently available at <http://www.opencontent.org/openpub/>).

The information contained herein is subject to change without notice. Red Hat, Inc. shall not be liable for technical or editorial errors or omissions contained herein.

Distribution of modified versions of this document is prohibited without the explicit permission of Red Hat Inc.

Distribution of this work or derivative of this work in any standard (paper) book form for commercial purposes is prohibited unless prior permission is obtained from Red Hat Inc.

The GPG fingerprint of the security@redhat.com key is:
CA 20 86 86 2B D6 9D FC 65 F6 EC C4 21 91 80 CD DB 42 A6 0E



Comments and Feedback

In the spirit of open source, we invite anyone to provide feedback and comments on any reference architectures. Although we review our papers internally, sometimes issues or typographical errors are encountered. Feedback allows us to not only improve the quality of the papers we produce, but allows the reader to provide their thoughts on potential improvements and topic expansion to the papers.

Feedback on the papers can be provided by emailing refarch-feedback@redhat.com. Please refer to the title within the email.

Staying In Touch

Join us on some of the popular social media sites where we keep our audience informed on new reference architectures as well as offer related information on things we find interesting.

Like us on Facebook:

<https://www.facebook.com/rhrefarch>

Follow us on Twitter:

<https://twitter.com/RedHatRefArch>

Plus us on Google+:

<https://plus.google.com/u/0/b/114152126783830728030/>



Table of Contents

1 Executive Summary.....	1
2 Technologies Overview.....	2
2.1 Red Hat Enterprise Virtualization.....	2
2.1.1 Red Hat Enterprise Virtualization Hypervisor.....	3
2.2 Red Hat Storage.....	4
2.3 OpenShift Enterprise.....	5
2.3.1 Broker.....	6
2.3.2 Node.....	7
3 Reference Architecture Environment.....	8
4 Dynamic Name Server (DNS).....	12
4.1 System Configuration.....	12
4.1.1 Networking.....	12
4.1.2 Updates.....	12
4.1.3 Time Synchronization.....	13
4.2 named Service Configuration.....	13
4.2.1 Installing and Configuring the named Service.....	13
4.2.2 Configuring Zones for Dynamic Updates.....	14
4.2.3 Dynamically Updating Entries.....	18
5 Red Hat Storage Server.....	21
5.1 Installation.....	21
5.2 System Configuration.....	21
5.2.1 Network Configuration.....	21
5.2.2 Updates.....	22
5.2.3 Time Synchronization.....	23
5.2.4 Reboot.....	23
5.3 Storage Configuration.....	24
5.3.1 RAID Utility.....	24
5.3.2 Prepare Storage.....	31
5.3.3 Gluster.....	36
5.3.4 Firewall.....	39
5.3.5 Tuning.....	40



6 Red Hat Enterprise Virtualization Servers.....	41
6.1 Red Hat Enterprise Virtualization Hypervisor.....	41
6.1.1 Configuration.....	41
6.2 Red Hat Enterprise Linux Hypervisor.....	42
6.2.1 Network Configuration.....	42
6.2.2 Updates.....	43
6.2.3 Time Synchronization.....	43
6.2.4 Storage Client.....	43
6.2.5 SSH Configuration.....	44
6.3 Red Hat Enterprise Virtualization Manager.....	44
6.3.1 Network Configuration.....	44
6.3.2 Updates.....	45
6.3.3 Time Synchronization.....	45
6.3.4 Red Hat Enterprise Virtualization Manager Configuration.....	46
7 Red Hat Enterprise Virtualization Environment.....	48
7.1 Data Center Configuration.....	48
7.2 Cluster Configuration.....	48
7.3 Hypervisor Configuration.....	49
7.3.1 Red Hat Enterprise Virtualization Hypervisor.....	49
7.3.2 Red Hat Enterprise Linux Hypervisor.....	50
7.3.3 Power Management.....	51
7.4 Network Configuration.....	52
7.5 Storage Domain Configuration.....	53
8 OpenShift Enterprise.....	54
8.1 Broker Host.....	54
8.1.1 Network Configuration.....	54
8.1.2 Updates.....	55
8.1.3 Time Synchronization.....	59
8.1.4 MongoDB.....	59
8.1.5 ActiveMQ.....	61
8.1.6 MCollective.....	64
8.1.7 Broker application.....	65
8.1.7.1 Broker Plugins.....	70
8.1.7.2 httpd Authentication.....	71
8.1.7.3 Bundler.....	71



8.2 Node Host.....	73
8.2.1 Network Configuration.....	73
8.2.2 Updates.....	74
8.2.3 Time Synchronization.....	75
8.2.4 MCollective.....	75
8.2.5 Node Packages.....	76
8.2.6 Cartridges.....	76
8.2.7 SSH Configuration.....	77
8.2.8 Service Configuration.....	78
8.2.9 PAM Configuration.....	78
8.2.10 Cgroup Configuration.....	79
8.2.11 Disk Quotas.....	80
8.2.12 SELinux.....	82
8.2.13 Tuning the Kernel.....	83
8.2.14 Port Proxy Configuration.....	84
8.2.15 Node Settings.....	84
8.2.16 Facter Database.....	85
8.2.17 Reboot and Test.....	85
9 Creating Gears.....	86
10 High Availability of Node Host.....	92
10.1 Preparing the host node for failover.....	93
10.1.1 Prepare Shared Storage.....	93
10.1.1.1 Partition and format the shared storage.....	94
10.1.2 Move Directories to Shared Storage.....	96
10.1.3 Copy Files and Other Information to Shared Storage.....	101
10.1.3.1 Pam Limits.....	102
10.1.3.2 User IDs and Control Groups.....	102
10.1.3.3 Cron Job.....	103
11 Failing the Node Host.....	105
11.1 Backup Node Host Installation And Configuration.....	105
11.1.1 Network Configuration.....	105
11.1.2 Time Synchronization.....	105
11.1.3 Updates.....	105
11.2 Power Off the Failed Node Host.....	109
11.3 Power On the Backup Node Host	110
11.3.1 Attach Shared Storage.....	110



11.3.2 Mount File Systems.....	110
11.3.3 Update Networking.....	111
11.3.4 Create User Accounts.....	111
11.3.5 Cron Configuration.....	111
11.3.6 Factor Database.....	112
11.3.7 Enable Services and Reboot.....	113
11.4 Testing.....	113
12 Failing Red Hat Storage Nodes.....	115
12.1 Verify Applications.....	115
12.2 Create an Application.....	115
12.3 Healing the Storage Files.....	117
13 Conclusion.....	119
Appendix A: Resources.....	120
Appendix B: Scripts.....	121
B.1 roses-common.sh.....	121
B.2 roses-ns.sh	124
B.3 roses-rhss.sh	128
B.4 roses-rhs-srv3.cfg	131
B.5 roses-rhs-srv4.cfg	131
B.6 roses-rhelh.sh	131
B.7 roses-rhevm.sh	133
B.8 roses-broker.sh	141
B.9 roses-node.sh	146
B.10 roses-backup-node.cfg	151
B.11 roses-create-app.sh	152
B.12 roses-node-prepare-shared.sh	154
B.13 roses-node-failover.sh	157
Appendix C: Contributors.....	161
Appendix D: Revision History.....	162



1 Executive Summary

Deploying applications in an OpenShift Enterprise environment allows developers to quickly test new software applications. After the applications are tested, they can quickly and easily be placed into production.

Failure of a production or developer applications can result in lost time and money. These losses can easily add to costs and expenses. The loss of a production application can affect end users or customer, further affecting cost and expenses as well as the impression of applications reliability.

Chances of application and data loss can be reduced by utilizing Red Hat Enterprise Virtualization and Red Hat Storage Server to run the OpenShift Enterprise platform. Using Red Hat Storage Server to provide the storage backend for the virtual machine storage and the Red Hat Storage Servers replication feature, data is stored on multiple hosts in the event of a storage failure.

Utilizing the High Availability features built into Red Hat Enterprise Virtualization, any brokers or nodes for the OpenShift Enterprise platform that run as virtual machines can be automatically moved to a different hypervisor if the hypervisor they are currently running on fails. However, this does not protect against a failure of the operating system within the virtual machine. A failure within the virtual machine can result in data loss if the failed virtual machine is an OpenShift Enterprise platform node host running user applications or gears.

Configuring the OpenShift Enterprise platform node host to use shared storage to store the data and configuration files allows for quick failover of a host node.

The purpose of this reference architecture is to demonstrate deploying OpenShift Enterprise platform in a highly available state using Red Hat Enterprise Virtualization and Red Hat Storage Server. Red Hat Storage Server is used as the backend storage to contain the virtual machine disk files. Red Hat Storage Server will be configured to replicate the stored data across multiple storage nodes.

The OpenShift Enterprise platform Node hosts running the user applications or gears will be virtual machines. The configuration and application data will be written to a shared virtual disk that can be migrated between two virtual machines in the case of a virtual machine failure.



2 Technologies Overview

2.1 Red Hat Enterprise Virtualization

Virtualization offers tremendous benefits for enterprise IT organizations – server consolidation, hardware abstraction, and internal clouds deliver a high degree of operational efficiency.

Red Hat Enterprise Virtualization (RHEV) combines the KVM hypervisor (powered by the Red Hat Enterprise Linux kernel) with an enterprise grade, multi-hypervisor management platform that provides key virtualization features such as live migration, high availability, power management, and virtual machine life cycle management. Red Hat Enterprise Virtualization delivers a secure, robust virtualization platform with unmatched performance and scalability for Red Hat Enterprise Linux and Windows guests.

Red Hat Enterprise Virtualization consists of the following two components:

- *RHEV MANAGER (RHEV-M)*: A feature-rich virtualization management system that provides advanced capabilities for hosts and guests.
- *RHEV HYPervisor*: A modern, scalable, high performance hypervisor based on RHEL KVM. It can be deployed as RHEV-H, a small footprint secure hypervisor image included with the RHEV subscription, or as a RHEL server (purchased separately) managed by RHEV-M.

A *HOST* is a physical server which provides the CPU, memory, and connectivity to storage and networks that are used for the virtual machines (VM). The local storage of the standalone host is used for the RHEV-H executables along with logs and enough space for ISO uploads.

A *CLUSTER* is a group of hosts of similar architecture. The requirement of similar architecture allows a virtual machine to be migrated from host to host in the cluster without having to shut down and restart the virtual machine. A cluster consists of one or more hosts, but a host can only be a member of one cluster.

A *DATA CENTER* is a collection of one or more clusters that have resources in common. Resources that have been allocated to a data center can be used only by the hosts belonging to that data center. The resources relate to storage and networks.

A *STORAGE DOMAIN* is a shared or local storage location for guest image files, import/export or for ISO images. Storage domain types supported in RHEV 3.0 are NFS, iSCSI, Fibre Channel, and local disk storage.

The RHEV *NETWORK* architecture supports both guest traffic and traffic among RHEV hypervisors and the RHEV-M server. All hosts have a network interface assigned to the logical network named *rhevm*. This network is used for the communications between the hypervisor and the manager. Additional logical networks are created on the data center and applied to one or more clusters. To become operational, the host attaches an interface to the local network. While the actual physical network can span across data centers, the logical network can only be used by the clusters and hosts of the creating data center.



2.1.1 Red Hat Enterprise Virtualization Hypervisor

A hypervisor is a computer software platform that allows multiple “guest” operating systems to run concurrently on a host computer. The guest virtual machines interact with the hypervisor which translates guest I/O and memory requests into corresponding requests for resources on the host computer.

Red Hat Enterprise Virtualization uses the Kernel-based Virtual Machine (KVM)¹, which turns Linux into a hypervisor. Red Hat Enterprise Linux 5.4 provided the first commercial-strength implementation of KVM, which is developed as part of the upstream Linux community. RHEV 3.0 uses the RHEL 6 KVM hypervisor, and inherits performance, scalability and hardware support enhancements from RHEL 6.

Figure 2.1.1.1: Typical RHEV Environment provides a graphical representation of a typical Red Hat Enterprise Virtualization environment with each component listed.

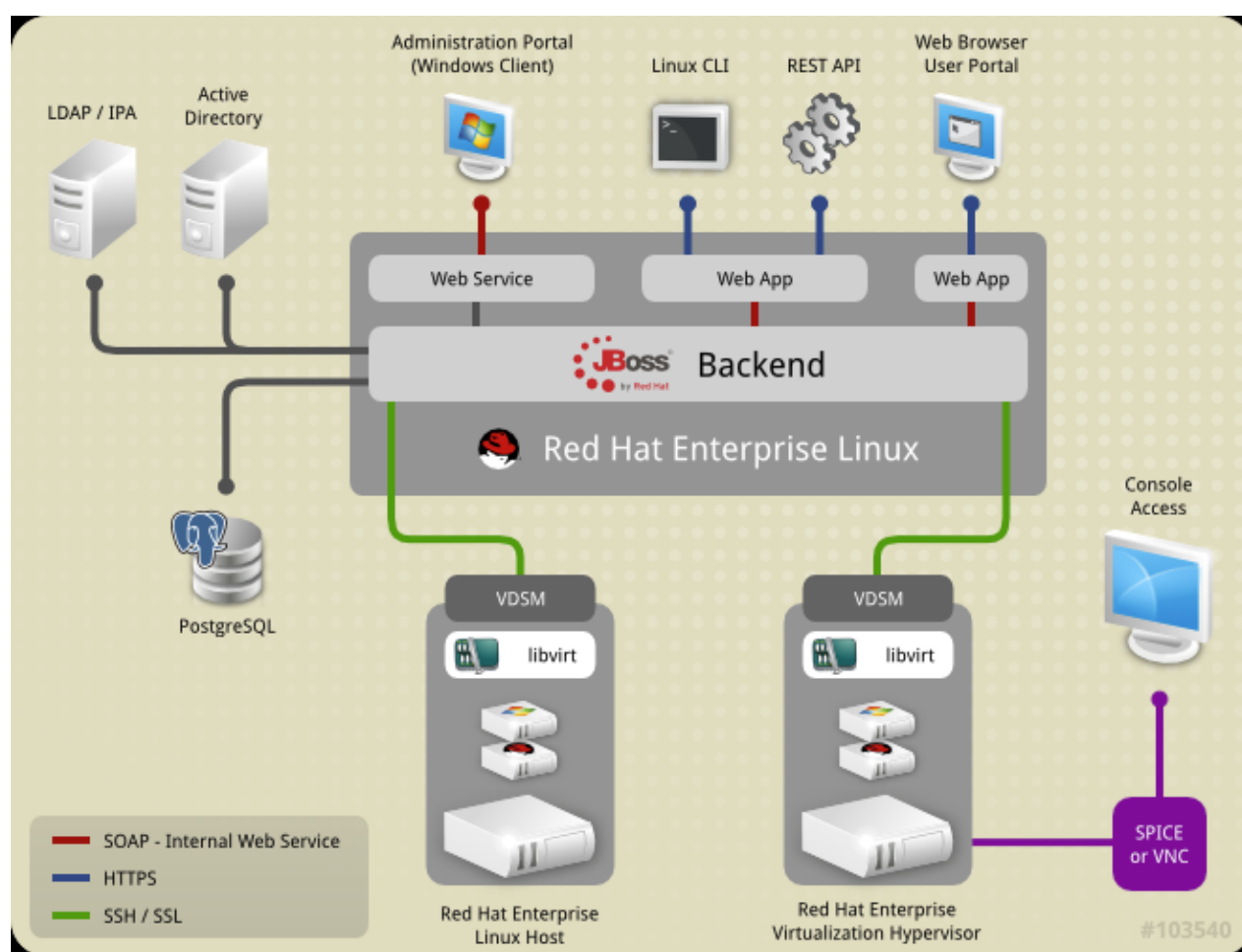


Figure 2.1.1.1: Typical RHEV Environment

¹ <http://www.redhat.com/promo/qumranet/>



2.2 Red Hat Storage

Red Hat Storage (RHS) is a scale-out network attached storage (NAS) for private cloud or data center, public cloud, and hybrid cloud environments. It is software-only, open source, and designed to meet unstructured data storage requirements. It enables enterprises to combine large numbers of commodity storage and compute resources into a high-performance, virtualized, and centrally managed storage pool. Both capacity and performance can scale linearly and independently on-demand, from a few terabytes to petabytes and beyond, using both on-premise commodity hardware and the public cloud compute/storage infrastructure. By combining commodity economics with a scale-out approach, organizations can achieve radically better price and performance in an easily deployed and managed solution that can be configured for increasingly demanding workloads. Red Hat Storage is built using the GlusterFS open source project as the foundation.

Red Hat Storage provides the following features and capabilities:

Linear Scaling and Scale-out Architecture – Red Hat Storage is built around multiple dimension scaling model imparting flexibility to suit different demands. The capacity can be scaled up or down in a single dimension by just adding/removing one component (disks or CPU or IO) or in multiple dimensions by simultaneously adding/removing multiple components (disks or CPU or IO), achieved by adding or removing storage nodes.

Bare Metal and Cloud - Red Hat Storage Server can be deployed on bare metal machines, in the private cloud, or on a public cloud like Amazon EC2. It also possible to combine machines across public and private clouds into a storage pool.

Software-Only - Red Hat Storage is delivered as a virtual appliance, either packaged within an ISO or an image deployed in a public cloud or as a package bundle that can be deployed on an existing Red Hat Enterprise Linux server. Red Hat Storage operates in user space.

Elasticity - Red Hat Storage allows enterprises to add or delete users, application data, volumes and storage nodes without disrupting any running functionality within the infrastructure.

Decentralized with No Metadata - Unlike other storage systems with a distributed file system, Red Hat Storage does not create, store, or use a separate index of metadata in any way and instead uses Elastic Hash Algorithm, rendering a true scale out capability. The 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 – Red Hat Storage supports data replication locally (N-Way Local Synchronous Replication) or over long distance (Geo-Replication asynchronous). It also supports replication in the private cloud/data center, public cloud or hybrid cloud environments using both N-Way and Geo-Replication methods.

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



Figure 2.2.1: Red Hat Storage Architecture depicts the architecture of a Red Hat Storage Solution.

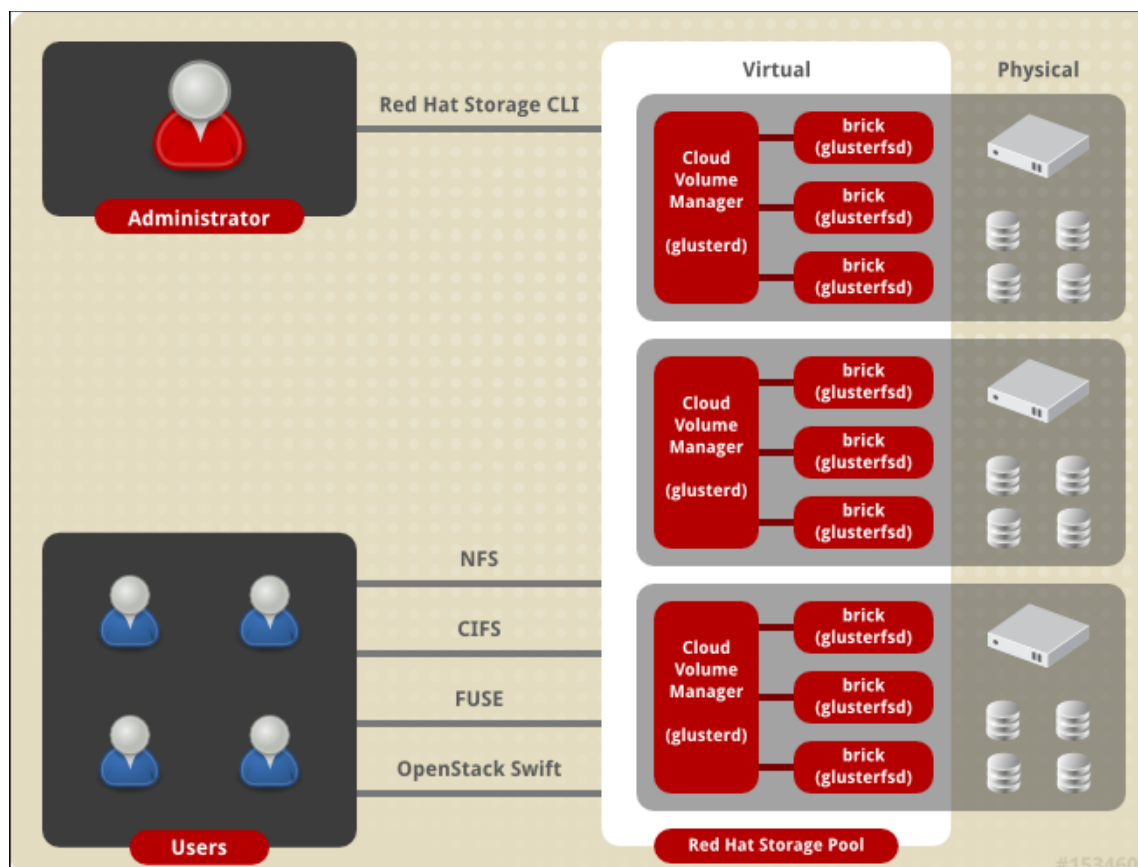


Figure 2.2.1: Red Hat Storage Architecture

2.3 OpenShift Enterprise

A Platform-as-a-Service, or PaaS, is a cloud application platform in which the application configuration, build, deployment, hosting, and administration is automated in an elastic cloud environment along with their supported stacks. OpenShift is a Platform as a Service (PaaS) solution from Red Hat. OpenShift provides a multi-language, auto-scaling, self-service, elastic cloud application platform built on a proven stack of Red Hat technologies. The OpenShift PaaS services are available as both a hosted service and on-premise PaaS product offering. OpenShift Enterprise is Red Hat's on-premise PaaS software solution that enables customers to deploy their own Private or Hybrid PaaS environment.

OpenShift Enterprise enables administrators to more quickly serve the needs of application developers by deploying a PaaS platform that streamlines the application service delivery process. OpenShift Enterprise enables administrators and developers to more quickly serve their needs by standardizing and accelerating developer work-flows. OpenShift Enterprise does this in a highly-efficient, fine-grained, multi-tenant cloud architecture that maximizes



infrastructure utilization. This provides application developers with self-service access so they can easily deploy applications on-demand leveraging the languages, frameworks, and tools of their choosing. It ultimately increases developer efficiency and agility by allowing them to focus on what they do best, writing code, and in turn enables them to better meet the demands of the business.

OpenShift Enterprise is built on the strength of a proven stack of Red Hat technologies. It provides the freedom for developers and administrators to work the way they want to work. For developers, OpenShift Enterprise supports Java, Ruby, PHP, Python, and Perl programming languages with associated frameworks as well as additional platform services like MySQL and PostgreSQL databases, Jenkins Continuous Integration server, and more. OpenShift Enterprise "cartridges" provide these platform services. Cartridges are extensible, enabling customers to extend OpenShift to add their own custom services. Developers can access OpenShift Enterprise via a rich command line interface (CLI), web console, RESTful API, or Eclipse integrated development environment (IDE). For IT administrators, OpenShift Enterprise runs on the trusted Red Hat Enterprise Linux platform and leverages technologies such as SELinux and Cgroups to provide a secure and scalable multi-tenant environment, that allows efficient infrastructure utilization supporting application development and deployment. OpenShift also provides standards-based application run-time systems like JBoss Enterprise Application Platform, Tomcat, and Apache. OpenShift Enterprise can be deployed on-premise in a customer's data center or private cloud on their choice of virtualization platform or cloud provider. In addition, developers can work from a variety of workstations including Red Hat Enterprise Linux, Mac OS X and Microsoft Windows. OpenShift Enterprise is open source, based on the upstream OpenShift Origin project, that helps drive innovation and eliminate lock-in. For more information on OpenShift Enterprise, please see the product documentation².

2.3.1 Broker

The *broker* is the single point of contact for all application management activities. It is responsible for the following tasks:

- Manage user authentication
- DNS updates
- Application state
- Application orchestration
- Services and operations

In addition to being able to contact the *broker* directly via a RESTful API, the following methods may also be used

- Web console
- Command line interface (CLI) tools
- JBoss Developer Studio

² https://access.redhat.com/site/documentation/OpenShift_Enterprise/



2.3.2 Node

An OpenShift *node* is a host that runs the applications. There can be many nodes that are deployed that a broker manages. A *node* supports gears that contain applications. Gears represent the system resources that an application is consuming. One of the features of OpenShift is that it provides a true multi-tenant environment. This is accomplished by using SELinux and Cgroup restrictions to separate each applications' resources and data and also providing quality of service controls.



3 Reference Architecture Environment

The environment used in this reference architecture is depicted in **Figure 3.1: Reference Architecture Environment**. This environment consists of four physical servers and five virtual machines. The roses-rhevm and roses-ns virtual machines run on a Red Hat Enterprise Linux 6 KVM server outside the Red Hat Enterprise Virtualization environment. The host names used in this paper are meant to be descriptive of their function. The word *ROSES* in the name of the hosts is a code name used for this reference architecture to differentiate the hosts from other hosts used for other projects within the project lab.

The Red Hat Enterprise Virtualization environment consists of two hypervisors, roses-rhelh and roses-rhev and the Red Hat Enterprise Virtualization Manager running on the roses-rhevm virtual machine. The storage for the Red Hat Enterprise Virtualization environment is provided by two Red Hat Storage Servers , ra-rhs-srv3 and ra-rhs-srv4. Storage traffic is isolated on its own network separate of the main public traffic.

The environment is configured with a data center called *DC-MAIN*. This data center consists of the CL-Production cluster and the *DATA-DL-MAIN* data domain. The cluster contains the two hypervisors, roses-rhelh and roses-rhev. The *DATA-DL-MAIN* data domain is backed by the Red Hat Storage Servers which replicate the data between them for redundancy. Two networks are present, *RHEVM* and *RHSTORAGE*. The *RHEVM* network is connected to the public network and is used for virtual machine traffic. The *RHSTORAGE* network is used by the hypervisors for storage traffic to the Red Hat Storage Servers.



Three virtual machines are present in the environment. Each virtual machine is created identically and all run Red Hat Enterprise Linux 6. A virtual disk is present that is shared between the roses-node and roses-backup-node. This virtual disk contains the gear data and other configuration necessary to fail the roses-node over to the roses-backup-node.

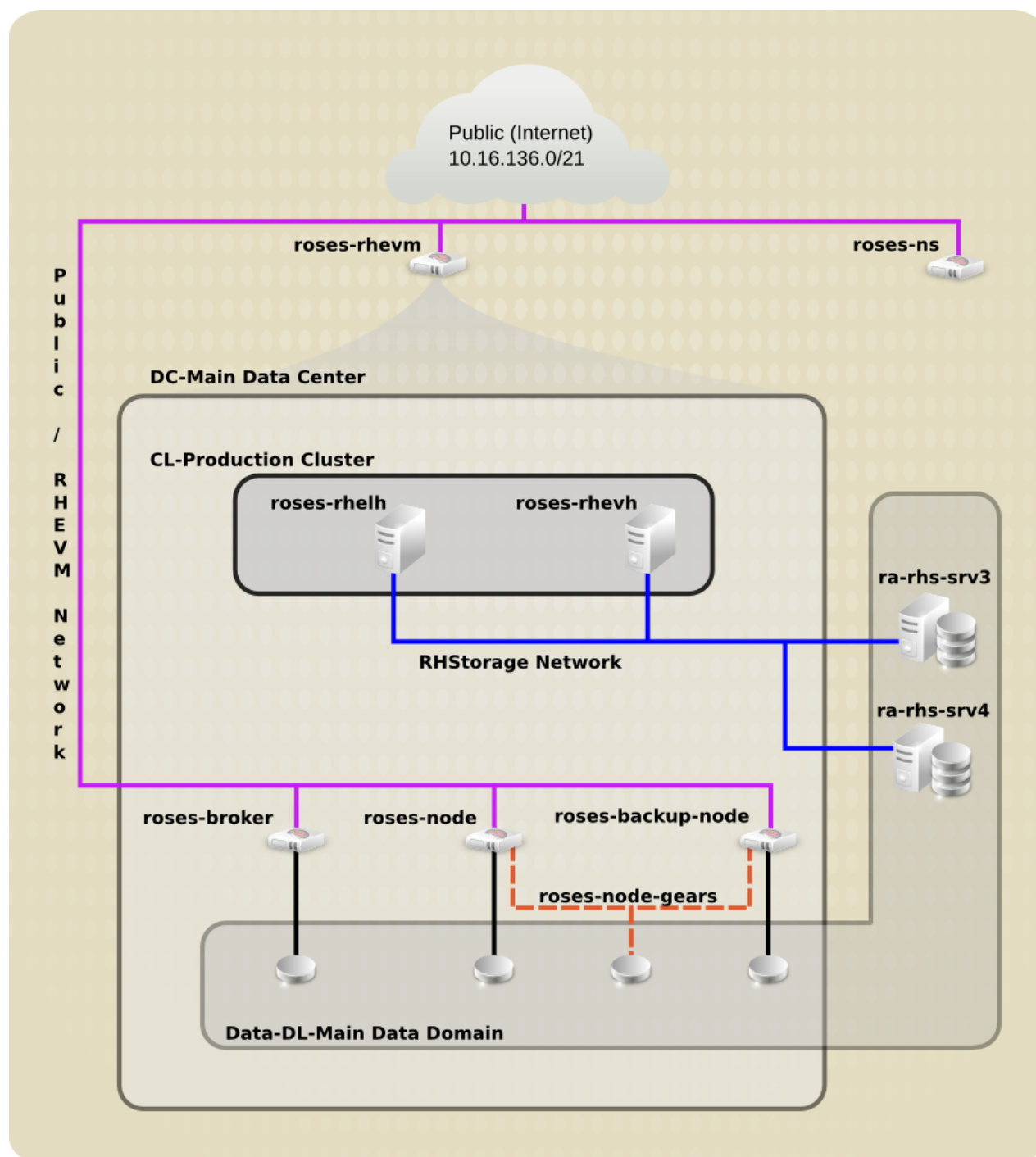


Figure 3.1: Reference Architecture Environment



The following table lists the configuration of the various servers.

Server	Configuration
roses-rhevm [KVM Virtual Machine]	RHEL 6.4 kernel 2.6.32-358.el6.x86_64
	SELinux Targeted Enforcing selinux-policy-3.7.19-195.el6_4.12.noarch
	rhevm-3.2.3-0.43.el6ev
	1 x QEMU Virtual CPU version (cpu64-rhel6) @ 2.4 GHz
	3 GB Memory
	1 x VirtIO Disk Device backed by logical volume @ 10 GB
	1 x VirtIO Network Adapter
roses-ns [KVM Virtual Machine]	RHEL 6.4 kernel 2.6.32-358.18.1.el6.x86_64
	SELinux Targeted Enforcing selinux-policy-3.7.19-195.el6_4.12.noarch
	bind-9.8.2-0.17.rc1.el6_4.6.x86_64
	1 x QEMU Virtual CPU version (cpu64-rhel6) @ 2.4 GHz
	1 GB Memory
	1 x VirtIO Disk Device backed by logical volume @ 200 GB
	1 x VirtIO Network Adapter
roses-rhelh [HP Proliant BL460c G6]	RHEL 6.4 kernel 2.6.32-358.23.2.el6.x86_64
	2 x Quad Core Intel Xeon CPU X550 @ 2.67GHz
	48 GB Memory
	2 x 146 GB SAS Internal Disk Drives - RAID 1
	3 x Broadcom NetExtreme II BCM57711E Flex-10 10GB Ethernet Controller
roses-rhevh [HP Proliant BL460c G6]	RHEV Hypervisor 6.4-20130912.1.el6_4
	2 x Quad Core Intel Xeon CPU X550 @ 2.67GHz
	48 GB Memory
	2 x 146 GB SAS Internal Disk Drives - RAID 1
	3 x Broadcom NetExtreme II BCM57711E Flex-10 10GB Ethernet Controller



Server	Configuration
ra-rhs-srv3 / ra-rhs-srv4 [Dell PowerEdge R510]	Red Hat Storage Server 2.1 kernel 2.6.32-358.23.2.el6.x86_64
	2 x Six Core Intel Xeon CPU X5650 @ 2.67GHz
	48 GB Memory
	2 x 500 GB SAS Internal Disk Drives - RAID 1 6 x 500 GB SAS Internal Disk Drives - RAID 6
	2 x Broadcom NetExtreme II BCM5716 Gigabit Ethernet Controller
	2 x Intel 82599EB 10-Gigabit SFI/SFP+ Ethernet Controller

Table 3.1: Server Configuration

The following table lists the configuration used to create the virtual machines in the Red Hat Enterprise Virtualization environment.

Tab	Option	Value
General	Data Center	DC-Main
	Host Cluster	CL-Production
	Memory Size	1024 MB
	Total Virtual CPUs	1
	Operating System	Red Hat Enterprise Linux 6.x x64
High Availability	Highly Available	Checked
	Priority for Run/Migration queue	High
Resource Allocation	Physical Memory Guaranteed	1024 MB
Virtual Disk	Size	12 GB
	Is bootable	Checked
Network Interface	Name	roses
	Network	rhevm
	Link State	Up

Table 3.2: OSE Virtual Machine Configuration



4 Dynamic Name Server (DNS)

Name resolution and dynamic updates are required for an OpenShift Enterprise environment to function properly. This section configures the roses-ns.example.org host to provide name resolution services using BIND.

4.1 System Configuration

4.1.1 Networking

Install a system with the latest version of Red Hat Enterprise Linux 6³. After the system is installed, configure the networking as listed in .

Option	Value
Hostname	roses-ns.example.org
IP Address	10.16.136.36
Network Mask	255.255.248.0
Gateway	10.16.143.254
DNS Server	10.16.136.36 10.16.143.248

Table 4.1.1.1: roses-ns Network Configuration

4.1.2 Updates

Register the system to receive updates and ensure the packages on the system are at the latest available versions.

```
# yum -y update  
[ ... Output Abbreviated ... ]
```

³ https://access.redhat.com/site/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Installation_Guide/index.html



4.1.3 Time Synchronization

Failure to have the correct time set on all hosts can result in loss of data or database entries that are out of order. The **ntpd** daemon is used to ensure the time on the hosts are correct. If the clock on the host is too far out of sync, then the **ntpd** daemon cannot make the initial adjustment. The initial adjustment must be made manually using the **date** command.

Enable and start the **ntpd** service to ensure the system has the correct time.

```
# date                                # Incorrect time
Wed Oct 23 09:30:09 CDT 2013

# date 10231127                        # Correct time set to 10 (October), 23rd, 11:27
Wed Oct 23 11:27:00 CDT 2013

# chkconfig ntpd on

# service ntpd start

Starting ntpd:                        [ OK ]
```

4.2 named Service Configuration

4.2.1 Installing and Configuring the named Service

Install the **bind** and **bind-utils** packages. These supply the daemon and utilities for name resolution.

```
# yum -y install bind bind-utils

[ ... Output Abbreviated ... ]
```

The **named** service is configured to provide name resolution and dynamic updates for the *example.org* network. Name resolution requests are forwarded for systems not on its network to the 10.16.143.247 and 10.16.143.248 name servers for name resolution.

The main configuration file for the named service is */etc/named.conf*. This file contains an **include** directive that tells the named service the configuration is in multiple files.

Create a backup copy of the */etc/named.conf* before editing the file.

```
# cp /etc/named.conf{,.orig}
```



Create the `/etc/named.conf` file with the following contents.

```
options {
    listen-on port 53 { any; };
    directory      "/var/named";
    dump-file      "/var/named/data/cache_dump.db";
    statistics-file "/var/named/data/named_stats.txt";
    memstatistics-file "/var/named/data/named_mem_stats.txt";
    allow-query    { any; };
    recursion yes;
    forwarders { 10.16.143.248 port 53; 10.16.143.247 port 53; };

    dnssec-enable no;
    dnssec-validation no;
    dnssec-lookaside auto;

    /* Path to ISC DLV key */
    bindkeys-file "/etc/named.iscdlv.key";

    managed-keys-directory "/var/named/dynamic";
};

logging {
    channel default_debug {
        file "data/named.run";
        severity dynamic;
    };
};

zone "." IN {
    type hint;
    file "named.ca";
};

include "/etc/named.rfc1912.zones";
include "/etc/named.root.key";
include "/etc/named.roses.zones";
```

4.2.2 Configuring Zones for Dynamic Updates

The `named` service is also configured to allow systems using a DNS security key to provide name resolution updates to the service. This key must be included in the configuration file.

The **`dnssec-keygen`** command generates a public and private security key. The command can create several types of keys using several encryption algorithms. See the man page for the **`dnssec-keygen`** command for more information. A key type of **`USER`** is generated for keys used to update name resolution entries. This is specified using the **`-n USER`** option. A 256 bit key is generated using the HMAC-MD5 algorithm. These are specified using the **`-b 256`** and **`-a hmac-md5`** options. The last option supplied to the command is the name of the key to generate. The zone name **`example.org`** is used for the name of this key.



Generate the security key in the */var/named* directory.

```
# cd /var/named  
  
# dnssec-keygen -a hmac-md5 -b 256 -n USER example.org  
  
Kexample.org.+157+38086
```

The prefix for the key files generated is displayed on the output. Use the **ls** command to display the key files.

```
# ls K*  
  
Kexample.org.+157+38086.key  Kexample.org.+157+38086.private
```

The file ending with *.key* is the public key file and the file ending with *.private* is the private key file. View the key files.

```
# cat Kexample.org.+157+38086.key  
  
example.org. IN KEY 0 3 157 GpqmTSUeLes1/x8FPQnftdTd4mmzoRPjtLOA6s5ciK0=  
  
# cat Kexample.org.+157+38086.private  
  
Private-key-format: v1.3  
Algorithm: 157 (HMAC_MD5)  
Key: GpqmTSUeLes1/x8FPQnftdTd4mmzoRPjtLOA6s5ciK0=  
Bits: AAA=  
Created: 20130829153601  
Publish: 20130829153601  
Activate: 20130829153601
```

The */etc/named.conf* file contains an *INCLUDE "/etc/named.roses.zones"*. This directive instructs the named service to import, or append, the configuration stored in the */etc/named.roses.zones* file into the */etc/named.conf* file. The */etc/named.roses.zones* file contains the directives that define the zones the named service provides resolution for. This file also contains a directive defining the key needed to update the name resolution records.



Create the file called `/etc/named.roses.zones` with the following contents.

```
zone "example.org." IN {
    type master;
    notify yes;
    file "dynamic/example.org.zone";
    allow-update { key example.org; };
};

zone "136.16.10.in-addr.arpa." IN {
    type master;
    notify yes;
    file "dynamic/136.16.10.in-addr.arpa.zone";
    allow-update { key example.org; };
};

key example.org {
    algorithm hmac-md5;
    secret "Mthbr5e1M6LD8I2k jy3gti1Skfmd5Az0QwZZtR4ehhM=";
};
```

This file defines the forward lookup zone called **roses.example.org** and the reverse lookup zone called **136.16.10.in-addr.arpa**. The file also defines a key called **roses.example.org**. This key is used by the zones to check if an request to update a name resolution entry is valid.

Set the correct ownership and SELinux security contexts for the file.

```
# chown -v root:named /etc/named.roses.zones

# chcon --reference /etc/named.root.key /etc/named.roses.zones
```

Since the name resolution service is configured to allow dynamic updates, the zone configuration files are placed in the `/var/named/dynamic` directory. This directory is used because it has the correct permissions and SELinux security contexts⁴.

⁴ https://access.redhat.com/site/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Managing_Confined_Services/chap-Managing_Confined_Services-Berkeley_Internet_Name_Domain.html



Create the forward and reverse lookup zones for the **example.org** domain. Place entries for the physical hosts in the zone files.

/var/named/dynamic/example.org.zone:

```
$TTL 86400
example.org      IN SOA      roses-ns.example.org. root.roses-ns.example.org.
(
                2013082818
                28800
                7200
                604800
                86400
                )
                NS      roses-ns.example.org.

roses-ns         A      10.16.136.36
```

/var/named/dynamic/136.16.10.in-addr.arpa.zone:

```
$TTL 86400
136.16.10.in-addr.arpa.  IN SOA      roses-ns.example.org. root.roses-ns-
ns.example.org. (
                2013082802
                28800
                7200
                604800
                86400
                )
                NS      ns.roses.example.org.

36               PTR    roses-ns.example.org.
```

Make sure the zone files have the correct ownership.

```
# chown -v named.named /var/named/dynamic/136.16.10.in-addr.arpa.zone
# chown -v named.named /var/named/dynamic/example.org.zone
```

Start the **named** service and set it to start upon boot.

```
# service named start

Generating /etc/rndc.key:      [ OK ]
Starting named:                [ OK ]

# chkconfig named on
```

Configure the firewall to allow dns traffic.

```
# lokkit --service dns
```




4.2.3 Dynamically Updating Entries

The name resolution service is now configured, but contains no entries for the systems in the environment. The **nsupdate** command is used to add entries to the zone files using dynamic updates. The **nsupdate** command requires access to both the public and private security key pair to submit the updates. The **-k** option is used to specify the key. The key pair is in the */var/named* directory, the **nsupdate** command is executed from this directory.

The **nsupdate** command provides a shell that accepts several internal commands. The nsupdate shell commands used in this environment are listed below.

server	Specifies the IP address of the name resolution server.
update	Instructs the command to update a record in the zone files. Takes either add or delete as the first option.
show	Displays the commands that have been queued, but have not been sent to the name resolution server.
send	Sends the commands that have been queued to the name resolution server.
quit	Exits the nsupdate shell.

Adding entries is done with the **update add** command. The syntax of the line to add entries to the forward lookup zone and the reverse lookup zone are similar.

Forward zone:

update add hostFQDN. TimeToLive RecordType IPAddress

Reverse zone:

update add ReverseIPAddress.in-addr.arpa TimeToLive RecordType hostFQDN.

There is a period placed after the **hostFQDN**, this period is required. See the documentation on **BIND** for information on the significance of the period in the zone configuration files.

Adding an entry to the forward lookup zone uses a record type of **A**. Adding entries to the reverse lookup zone uses a record type of **PTR**. The **IPAddress** on a reverse zone has its octets reversed and **.in-addr.arpa** appended to it to signify it is a reverse zone entry.



Multiple entries may be added to a zone before the send command is issued, but the send command must be issued before adding entries to a different zone. Add entries to the forward and reverse zones for the physical servers used in this environment.

```
# nsupdate -k Kexample.org.+157+38086.key
> server 10.16.136.36
> update add roses-rhelh.example.org. 38400 A 10.16.136.79
> update add roses-rhevh.example.org. 38400 A 10.16.136.78
> update add roses-rhevm.example.org. 38400 A 10.16.136.31
> update add roses-broker.example.org. 38400 A 10.16.136.76
> update add roses-node.example.org. 38400 A 10.16.136.74
> show
Outgoing update query:
;; ->>HEADER<<- opcode: UPDATE, status: NOERROR, id:      0
;; flags:;; ZONE: 0, PREREQ: 0, UPDATE: 0, ADDITIONAL: 0
;; UPDATE SECTION:
roses-rhelh.example.org. 38400    IN      A       10.16.136.79
roses-rhevh.example.org. 38400    IN      A       10.16.136.78
roses-rhevm.example.org. 38400    IN      A       10.16.136.31
roses-broker.example.org. 38400    IN      A       10.16.136.76
roses-node.example.org. 38400    IN      A       10.16.136.74

> send
> update add 79.136.16.10.in-addr.arpa 38400 PTR roses-rhelh.example.org.
> update add 78.136.16.10.in-addr.arpa 38400 PTR roses-rhevh.example.org.
> update add 31.136.16.10.in-addr.arpa 38400 PTR roses-rhevm.example.org.
> update add 76.136.16.10.in-addr.arpa 38400 PTR roses-broker.example.org.
> update add 74.136.16.10.in-addr.arpa 38400 PTR roses-node.example.org.
> show
Outgoing update query:
;; ->>HEADER<<- opcode: UPDATE, status: NOERROR, id:      0
;; flags:;; ZONE: 0, PREREQ: 0, UPDATE: 0, ADDITIONAL: 0
;; UPDATE SECTION:
79.136.16.10.in-addr.arpa. 38400 IN      PTR    roses-rhelh.example.org.
78.136.16.10.in-addr.arpa. 38400 IN      PTR    roses-rhevh.example.org.
31.136.16.10.in-addr.arpa. 38400 IN      PTR    roses-rhevm.example.org.
76.136.16.10.in-addr.arpa. 38400 IN      PTR    roses-broker.example.org.
74.136.16.10.in-addr.arpa. 38400 IN      PTR    roses-node.example.org.

> send
> quit
```



The entries are stored in a journal file and are not view-able in the zone configuration files until the journal entries are written to the configuration files. This can be forced by using the **rndc** command to **freeze** zone updates and then **thawing** the updates again.

```
# rndc freeze; sleep 3; rndc thaw

# cat dynamic/example.org.zone

$ORIGIN .
$TTL 86400 ; 1 day
example.org      IN SOA      roses-ns.example.org. root.roses-ns.example.org.
(
    2013082877 ; serial
    28800      ; refresh (8 hours)
    7200       ; retry (2 hours)
    604800     ; expire (1 week)
    86400      ; minimum (1 day)
)
                NS      roses-ns.example.org.
$ORIGIN example.org.
$TTL 38400 ; 10 hours 40 minutes
roses-ns        A      10.16.136.36
roses-rhelh     A      10.16.136.79
roses-rhevh     A      10.16.136.78
roses-rhevm     A      10.16.136.31

# cat dynamic/136.16.10.in-addr.arpa.zone

$ORIGIN .
$TTL 86400 ; 1 day
136.16.10.in-addr.arpa IN SOA      roses-ns.example.org. root.roses-
ns.example.org. (
    2013082821 ; serial
    28800      ; refresh (8 hours)
    7200       ; retry (2 hours)
    604800     ; expire (1 week)
    86400      ; minimum (1 day)
)
                NS      roses-ns.example.org.
$ORIGIN 136.16.10.in-addr.arpa.
$TTL 38400 ; 10 hours 40 minutes
36              PTR      roses-ns.example.org.
31              PTR      roses-rhevm.example.org.
78              PTR      roses-rhevh.example.org.
79              PTR      roses-rhelh.example.org.
```



5 Red Hat Storage Server

Unless otherwise stated, the steps in this section must be performed on both the ra-rhs-srv3 and ra-rhs-srv4 hosts.

5.1 Installation

The **Red Hat Storage Server Installation Guide**⁵ contains the instructions to download and install the **Red Hat Storage Server**.

Download and install **Red Hat Storage Server**⁶ on the ra-rhs-srv3 and ra-rhs-srv4 servers. Use the information in **Table 5.1.1: RHSS Network Configuration** to configure basic networking on both hosts.

Configuration	ra-rhs-srv3	ra-rhs-srv4
em1	10.16.143.95/21	10.16.143.96/21
p1p1	172.31.143.95/16	172.31.143.96/16
Gateway	10.16.143.254	10.16.143.254
Name Resolution	10.16.143.247 10.16.143.248	10.16.143.247 10.16.143.248

Table 5.1.1: RHSS Network Configuration

5.2 System Configuration

5.2.1 Network Configuration

The environment in this reference architecture implements a separate network for storage traffic. A second network interface must be configured for this traffic.

The network device used for the storage network is `/dev/p1p1`⁷. Configure the device by editing the `/etc/sysconfig/network-scripts/ifcfg-p1p1` file. Use the appropriate IP address from **Table 5.1.1: RHSS Network Configuration** for the **IPADDR** variable.

`/etc/sysconfig/network-scripts/ifcfg-p1p1`:

```
DEVICE="p1p1"
NM_CONTROLLED="no"
ONBOOT="yes"
BOOTPROTO=none
IPADDR=172.31.143.95
NETMASK=255.255.0.0
```

⁵ https://access.redhat.com/site/documentation/en-US/Red_Hat_Storage/2.0/html/Installation_Guide/

⁶ <https://access.redhat.com/site/pages/468693#downloads-page>

⁷ <https://access.redhat.com/site/articles/53579>



Start the network interface.

```
# ifup p1p1
```

5.2.2 Updates

After the storage systems are installed, register them with **RHN Classic** to receive updates. As of the writing of this paper, a Red Hat Storage Server cannot register using the **Red Hat Subscription Manager**. Attempting to register a system with Subscription Manager returns the message “*TOO MANY CONTENT SETS FOR CERTIFICATE*” See the **Red Hat Knowledge Base article number 129003**⁸ for more information.

Register the system to RHN Classic using the **rhncfg** command. The **--use-eus-channel** option is used to register the system to the *EXTENDED UPDATE SUPPORT*⁹ channel. The **--username** and **--password** options specify the login credentials to use when authenticating with RHN Classic.

```
# rhncfg --use-eus-channel --username [User] --password '[Password]'
```

The system is now registered to RHN Classic. Use the **rhncfg** command with the **--list** option to display the channels the system is currently subscribed to.

```
# rhncfg --list
rhel-x86_64-server-6.4.z
```

The system must be subscribed to the channels listed in **Table 5.3.2.1: Virtual Drive Information** to receive updates for all the packages.

Channel Description	Channel Name
Red Hat Storage Server	rhel-x86_64-server-6-rhs-2.1
Red Hat Enterprise Linux Scalable File System	rhel-x86_64-server-sfs-6.4.z

Table 5.2.2.1: RHN Classic RHSS Channels

Use the **--add** and **--channel** options with the **rhncfg** command to subscribe the system to the appropriate channels.

```
# rhncfg --user admin --password "[Password]" --add --channel rhel-x86_64-server-6-rhs-2.1 --channel rhel-x86_64-server-sfs-6.4.z
```

Check if the system is subscribed to the appropriate channels.

```
# rhncfg --list
rhel-x86_64-server-6-rhs-2.1
rhel-x86_64-server-6.4.z
rhel-x86_64-server-sfs-6.4.z
```

⁸ <https://access.redhat.com/site/solutions/129003>

⁹ <https://access.redhat.com/site/solutions/22763>



Update the packages on the system to the latest versions.

```
# yum -y update
Loaded plugins: aliases, changelog, downloadonly, fastestmirror, filter-
data, keys, list-data, merge-conf, priorities, product-id, protectbase,
rhnplugin, security,
                : subscription-manager, tmprepo, tsflags, upgrade-helper,
verify, versionlock
Updating certificate-based repositories.
Loading mirror speeds from cached hostfile

[ ... Output Abbreviated ... ]

Installed products updated.

Installed:
[ ... Output Abbreviated ... ]

Updated:
[ ... Output Abbreviated ... ]

Complete!
```

5.2.3 Time Synchronization

The Red Hat Storage Servers need the correct times set on each server. Enable the **ntpd** service. Check the date and time of the system and make sure it is correct. Fix the date and time if necessary.

```
# chkconfig ntpd on
```

5.2.4 Reboot

Reboot the system to ensure the latest versions of the packages are running.

```
# init 6
```



5.3 Storage Configuration

5.3.1 RAID Utility

The output of the **lspci** command shows the RAID controller in the server is an LSI controller.

```
# lspci
00:00.0 Host bridge: Intel Corporation 5500 I/O Hub to ESI Port (rev 13)

[ ... Output Abbreviated ... ]

01:00.0 Ethernet controller: Broadcom Corporation NetXtreme II BCM5716
Gigabit Ethernet (rev 20)
01:00.1 Ethernet controller: Broadcom Corporation NetXtreme II BCM5716
Gigabit Ethernet (rev 20)
02:00.0 RAID bus controller: LSI Logic / Symbios Logic MegaRAID SAS 2108
[Liberator] (rev 05)
05:00.0 Ethernet controller: Intel Corporation 82599EB 10-Gigabit SFI/SFP+
Network Connection (rev 01)
05:00.1 Ethernet controller: Intel Corporation 82599EB 10-Gigabit SFI/SFP+
Network Connection (rev 01)
06:03.0 VGA compatible controller: Matrox Graphics, Inc. MGA G200eW WPCM450
(rev 0a)
```

Download and install the **MegaCLI** package from the LSI website www.lsi.com¹⁰.

```
# unzip 8.07.10_MegaCLI_Linux.zip
Archive: 8.07.10_MegaCLI_Linux.zip
  inflating: 8.07.10_MegaCLI_Linux/Linux MegaCLI 8.07.10.txt
    creating: 8.07.10_MegaCLI_Linux/Linux MegaCLI 8.07.10/
  inflating: 8.07.10_MegaCLI_Linux/Linux MegaCLI 8.07.10/MegaCli-8.07.10-
1.noarch.rpm

# yum -y localinstall 8.07.10_MegaCLI_Linux/Linux\ MegaCLI\ 8.07.10/MegaCli-
8.07.10-1.noarch.rpm

[ ... Output Abbreviated ... ]

Installed:
  MegaCli.noarch 0:8.07.10-1

Complete!
```

Once the package is installed, the **/opt/MegaRAID/MegaCli/MegaCli64** command is executed to display the configuration of the RAID devices. This command contains many options which are documented in various user guides available on the LSI website.

¹⁰ http://www.lsi.com/downloads/Public/MegaRAID%20Common%20Files/8.07.10_MegaCLI_Linux.zip



Use the **MegaCli64** program to view the controllers logical disk information. Use the following options to query the information.

- LDInfo** Display information about the virtual drives
- Lall** Display information for all virtual drives
- aALL** Include all controllers

```
# /opt/MegaRAID/MegaCli/MegaCli64 -LDInfo -Lall -aALL
```

```
Adapter 0 -- Virtual Drive Information:
```

```
Virtual Drive: 0 (Target Id: 0)
```

```
Name                    : rootdisk
```

```
RAID Level              : Primary-1, Secondary-0, RAID Level Qualifier-0
```

```
Size                    : 465.25 GB
```

```
Sector Size             : 512
```

```
Mirror Data             : 465.25 GB
```

```
State                   : Optimal
```

```
Strip Size              : 64 KB
```

```
Number Of Drives        : 2
```

```
Span Depth              : 1
```

```
Default Cache Policy: WriteBack, ReadAdaptive, Direct, No Write Cache if Bad BBU
```

```
Current Cache Policy: WriteBack, ReadAdaptive, Direct, No Write Cache if Bad BBU
```

```
Default Access Policy: Read/Write
```

```
Current Access Policy: Read/Write
```

```
Disk Cache Policy       : Disk's Default
```

```
Encryption Type         : None
```

```
Bad Blocks Exist: No
```

```
Is VD Cached: Yes
```

```
Cache Cade Type : Read Only
```

The output from the command indicates there is a single logical disk configured. The logical disk has a drive id of **0** and is named **rootdisk**. The logical disk is a mirrored array with a stripe size of **64KB**. The logical disk is in an **optimal** state.

Logical disk zero is a two drive **RAID 1** array. This array was previously configured using the controllers BIOS.



The **-CfgDsply** option can be used to display the physical disks that are members of logical disks. This command produces a lot of output. The **grep** command is used to display only the information needed to see physical disk membership. The output shows that logical disk 0 contains the physical drives in slots 12 and 13 of the enclosure with the ID of 32.

```
# /opt/MegaRAID/MegaCli/MegaCli64 -CfgDsply -aAll | grep -E "^DISK|^Virtual|^Name|^RAID|^Size|^Enclosure Device|^Slot|^Drive's position|^Physical Disk|^$"
```

```
DISK GROUP: 0
Virtual Drive Information:
Virtual Drive: 0 (Target Id: 0)
Name                : rootdisk
RAID Level           : Primary-1, Secondary-0, RAID Level Qualifier-0
Size                 : 465.25 GB
```

```
Physical Disk Information:
Physical Disk: 0
Enclosure Device ID: 32
Slot Number: 12
Drive's position: DiskGroup: 0, Span: 0, Arm: 0
```

```
Physical Disk: 1
Enclosure Device ID: 32
Slot Number: 13
Drive's position: DiskGroup: 0, Span: 0, Arm: 1
```

A new logical disk is created to store the data used in this reference architecture. Before creating the new logical disk, the number of unused physical disks must be known. The **-PDList** option is used to display the physical disks. The **grep** command is used to display only the relevant information.

```
# /opt/MegaRAID/MegaCli/MegaCli64 -PDList -aALL | grep -E 'Adapter|Enclosure Device|Slot|Raw Size|^$'
```

```
Adapter #0

Enclosure Device ID: 32
Slot Number: 0
Raw Size: 931.512 GB [0x74706db0 Sectors]

Enclosure Device ID: 32
Slot Number: 1
Raw Size: 931.512 GB [0x74706db0 Sectors]

Enclosure Device ID: 32
Slot Number: 2
Raw Size: 931.512 GB [0x74706db0 Sectors]

Enclosure Device ID: 32
Slot Number: 3
Raw Size: 931.512 GB [0x74706db0 Sectors]

Enclosure Device ID: 32
```



```
Slot Number: 4
Raw Size: 931.512 GB [0x74706db0 Sectors]

Enclosure Device ID: 32
Slot Number: 5
Raw Size: 931.512 GB [0x74706db0 Sectors]

Enclosure Device ID: 32
Slot Number: 6
Raw Size: 931.512 GB [0x74706db0 Sectors]

Enclosure Device ID: 32
Slot Number: 7
Raw Size: 931.512 GB [0x74706db0 Sectors]

Enclosure Device ID: 32
Slot Number: 8
Raw Size: 931.512 GB [0x74706db0 Sectors]

Enclosure Device ID: 32
Slot Number: 9
Raw Size: 931.512 GB [0x74706db0 Sectors]

Enclosure Device ID: 32
Slot Number: 10
Raw Size: 931.512 GB [0x74706db0 Sectors]

Enclosure Device ID: 32
Slot Number: 11
Raw Size: 931.512 GB [0x74706db0 Sectors]

Enclosure Device ID: 32
Slot Number: 12
Raw Size: 465.761 GB [0x3a386030 Sectors]

Enclosure Device ID: 32
Slot Number: 13
Raw Size: 465.761 GB [0x3a386030 Sectors]
```

The output shows there are physical disks in slots 0 through 13. The disks in slots 12 and 13 are used in a RAID 1 mirror and contain the operating system.



Only the first six of the remaining twelve drives are used to create the needed logical disk. The remaining disks are used for a different project or paper. The disks are assembled into a single logical disk using RAID 6. This RAID level was chosen for data redundancy purposes. The following options are provided to the **MegaCli** command to create the logical disk.

- CfgLdAdd** Add a logical disk to the configuration.
- r6 [32:0,32:1,32:2,32:3,32:4,32:5]**
 Create a RAID 6 (-r6) array using
 the first six disks (0-5) of
 enclosure 32.
- strpsz256** Use a stripe size of 256 KB for the array.
- a0** Create the logical disk using controller 0.

```
# /opt/MegaRAID/MegaCli/MegaCli64 -CfgLdAdd -r6
[32:0,32:1,32:2,32:3,32:4,32:5] -strpsz256 -a0
```

```
Adapter 0: Created VD 1
```

```
Adapter 0: Configured the Adapter!!
```

```
Exit Code: 0x00
```

Viewing the configuration now displays configuration for both the logical disks.

```
# /opt/MegaRAID/MegaCli/MegaCli64 -CfgDsply -aAll | grep -E "^DISK|^Virtual|^Name|^RAID|^Size|^Enclosure Device|^Slot|^Drive's position|^Physical Disk|^$"

```

DISK GROUP: 0

```
Virtual Drive Information:
```

```
Virtual Drive: 0 (Target Id: 0)
```

```
Name                       :rootdisk
```

```
RAID Level                 : Primary-1, Secondary-0, RAID Level Qualifier-0
```

```
Size                       : 465.25 GB
```

```
Physical Disk Information:
```

```
Physical Disk: 0
```

```
Enclosure Device ID: 32
```

```
Slot Number: 12
```

```
Drive's position: DiskGroup: 0, Span: 0, Arm: 0
```

```
Physical Disk: 1
```

```
Enclosure Device ID: 32
```

```
Slot Number: 13
```

```
Drive's position: DiskGroup: 0, Span: 0, Arm: 1
```

DISK GROUP: 1

```
Virtual Drive Information:
```

```
Virtual Drive: 1 (Target Id: 1)
```

```
Name                       :
```

```
RAID Level                 : Primary-6, Secondary-0, RAID Level Qualifier-3
```



```
Size                : 3.636 TB
Physical Disk Information:
Physical Disk: 0
Enclosure Device ID: 32
Slot Number: 0
Drive's position: DiskGroup: 1, Span: 0, Arm: 0

Physical Disk: 1
Enclosure Device ID: 32
Slot Number: 1
Drive's position: DiskGroup: 1, Span: 0, Arm: 1

Physical Disk: 2
Enclosure Device ID: 32
Slot Number: 2
Drive's position: DiskGroup: 1, Span: 0, Arm: 2

Physical Disk: 3
Enclosure Device ID: 32
Slot Number: 3
Drive's position: DiskGroup: 1, Span: 0, Arm: 3

Physical Disk: 4
Enclosure Device ID: 32
Slot Number: 4
Drive's position: DiskGroup: 1, Span: 0, Arm: 4

Physical Disk: 5
Enclosure Device ID: 32
Slot Number: 5
Drive's position: DiskGroup: 1, Span: 0, Arm: 5
```

The newly created logical disk does not contain a name. The **-LDSetProp** option can be used to set the name or other properties of the logical disk. The command uses the following options:

- LDSetProp** Set the properties of a logical disk.
- Name** Set the name property.
- L1** Apply the changes on logical volume 1.
- a0** Apply the changes on adapter 0.

Set the name of the logical disk to **ROSES**.

```
# /opt/MegaRAID/MegaCli/MegaCli64 -LDSetProp -Name ROSES -L1 -a0
Set name to ROSES on Adapter 0, VD 1 (target id: 1) success
Exit Code: 0x00
```



Viewing the information about the logical disk verifies the name has been set. The output shows the adapter is performing a background initialization on the logical disk.

```
# /opt/MegaRAID/MegaCli/MegaCli64 -LDInfo -L1 -aALL
```

```
Adapter 0 -- Virtual Drive Information:
```

```
Virtual Drive: 1 (Target Id: 1)
```

```
Name : ROSES
```

```
RAID Level : Primary-6, Secondary-0, RAID Level Qualifier-3
```

```
Size : 3.636 TB
```

```
Sector Size : 512
```

```
Parity Size : 1.818 TB
```

```
State : Optimal
```

```
Strip Size : 256 KB
```

```
Number Of Drives : 6
```

```
Span Depth : 1
```

```
Default Cache Policy: WriteBack, ReadAdaptive, Direct, No Write Cache if Bad BBU
```

```
Current Cache Policy: WriteBack, ReadAdaptive, Direct, No Write Cache if Bad BBU
```

```
Default Access Policy: Read/Write
```

```
Current Access Policy: Read/Write
```

```
Disk Cache Policy : Disk's Default
```

```
Ongoing Progresses:
```

```
Background Initialization: Completed 13%, Taken 15 min.
```

```
Encryption Type : None
```

```
Bad Blocks Exist: No
```

```
Is VD Cached: Yes
```

```
Cache Cade Type : Read Only
```

```
Exit Code: 0x00
```



5.3.2 Prepare Storage

The storage device must be managed by LVM and presented to the operating system as a logical volume. View the logical disk information using the **MegaCli** command.

```
# /opt/MegaRAID/MegaCli/MegaCli64 -LDInfo -L1 -aALL
```

Adapter 0 -- Virtual Drive Information:

Virtual Drive: 1 (Target Id: 1)

Name : ROSES

RAID Level : Primary-6, Secondary-0, RAID Level Qualifier-3

Size : 3.636 TB

Sector Size : 512

Parity Size : 1.818 TB

State : Optimal

Strip Size : 256 KB

Number Of Drives : 6

Span Depth : 1

Default Cache Policy: WriteBack, ReadAdaptive, Direct, No Write Cache if Bad BBU

Current Cache Policy: WriteBack, ReadAdaptive, Direct, No Write Cache if Bad BBU

Default Access Policy: Read/Write

Current Access Policy: Read/Write

Disk Cache Policy : Disk's Default

Encryption Type : None

Bad Blocks Exist: No

Is VD Cached: Yes

Cache Cade Type : Read Only

Exit Code: 0x00



Make note of the logical disks **Target Id**, **RAID Level**¹¹, **Stripe Size**, and **Number of Drives**. This information is used in the following steps when creating and formatting the storage. Providing this information performs some basic tuning during the creation and formatting operations. The values are listed in **Table 5.3.2.1: Virtual Drive Information**.

Configuration	Value
Target Id	1
RAID Level	6
Stripe Size	256 KB
Drives	6

Table 5.3.2.1: Virtual Drive Information

The **sg3_utils** package contains utilities to control and query devices that use the SCSI command sets. This is used to determine the system device name of the logical drive.

Execute the **sg_map** command that is contained in the package. Use the **-i** and **-x** options when executing the command. The **-i** option displays the device **INQUIRY** strings and the **-x** option shows device specific information. The output displays the **SCSI generic device name**, **bus**, **channel**, **target id**, **lun**, **type**, **system device name**, and the **INQUIRY** strings.

```
# sg_map -i -x
/dev/sg0 0 0 32 0 13 DP BACKPLANE 1.10
/dev/sg1 0 2 0 0 0 /dev/sda DELL PERC H700 2.10
/dev/sg2 0 2 1 0 0 /dev/sdb DELL PERC H700 2.10
/dev/sg3 1 0 0 0 0 /dev/sdc iDRAC LCDRIVE 0323
/dev/sg4 2 0 0 0 5 /dev/scd0 iDRAC Virtual CD 0323
/dev/sg5 2 0 0 1 0 /dev/sdd iDRAC Virtual Floppy 0323
```

The PERC H700 controller is the RAID controller in the system. Two devices are configured on the PERC H700. The fourth column of the output displays the target id for the device. The newly created logical drive has a target id of 1. The output shows target id 1 on the PERC H700 has a system device name of **/dev/sdb**.

View the disk information using the **fdisk** command to verify the size.

```
# fdisk -l /dev/sdb

Disk /dev/sdb: 3998.6 GB, 3998614552576 bytes
255 heads, 63 sectors/track, 486137 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000
```

¹¹ The format used to display the RAID level might seem confusing to some. See the Common RAID Disk Data Format Specification on the Storage Network Industry Associations (SNIA) website for an explanation of the output and RAID levels. http://www.snia.org/sites/default/files/SNIA_DDF_Technical_Position_v2.0.pdf



The logical disk must be added as a physical volume to the logical volume manager (LVM). The **pvcreate** command is used to add the device. This command can take several optional arguments as well as the device name. The **--dataalignment** argument is used to tell the **pvcreate** command to align the data to a multiple of the value specified. Specifying the data alignment allows the system to write chunks of data in a single stripe across the RAID disks.

This value is calculated by multiplying the number of active disks that contain data in the stripe by the stripe size of the RAID volume. Since the virtual drive has a RAID type of six, it contains four disks that contain data at any one time. The remaining two disks contain parity information and no data. Since the virtual disk has a stripe size of 256 KB, the calculation looks as follows:

$$\begin{array}{rclclcl} \text{Active Disks} & & \times & & \text{Stripe Size} & = & \text{Alignment} \\ 4 & & \times & & 256 \text{ KB} & = & 1024 \text{ KB} \end{array}$$

The value to specify for the **--dataalignment** option is 1024 KB. Add the */dev/sdb* disk to the LVM.

```
# pvcreate --dataalignment 1024KB /dev/sdb

Physical volume "/dev/sdb" successfully created
```

Use the **pvs** command to display the volume group information. The **-o** option specifies the column information to display. The **pv_name**, **vg_name**, **pv_size**, and **pe_start** options indicate the command should display the physical volume name, volume group name, physical volume size, and location of the first physical extent.

```
# pvs -o pv_name,vg_name,pv_size,pe_start --units k

PV          VG          PSize          1st PE
/dev/sda2   vg_rarhssrv3  487333888.00k  1024.00k
/dev/sdb    3904897024.00k 1024.00k
```

The output shows the data is aligned to the first 1024 KB.

Create a volume group on the new physical volume. The **--physicalextentsize** is used to increase the size of the extents to 64MB. This option is not necessary and the best value for the option is environment specific.

```
# vgcreate --physicalextentsize 64M ROSESvg /dev/sdb

Volume group "ROSESvg" successfully created
```

Create a one terabyte logical volume in the new volume group.

```
# lvcreate --size 1T --name datadomain_lv ROSESvg

Logical volume "datadomain_lv" created
```




The logical volume is formatted with an XFS file system. The size of the inodes must be increased to 512 bytes or greater. This is because Red Hat Storage uses extended attributes and the default inode size of 256 bytes is not large enough to hold them.

Other options may be specified on the command line to help tune the file system and increase performance. See the Red Hat Enterprise Linux Performance Tuning Guide¹² for more information. The options used in this environment and a short description of each are listed in

Table 5.3.2.2: XFS File System Creation Options.

Option	Argument	Description
-i	size=512	Create inodes of 512 bytes instead of the default 256 bytes. This is required to store the Red Hat Storage extended attributes.
	maxpct=0	Specify the amount of space on the file system to be used for inodes. Setting to 0, allows up to all the file system to be used for inodes if needed. This is set to 0 to prevent running out of inodes.
-d	su=256k	RAID device stripe size.
	sw=4	RAID device stripe width. This is the number of active disks in the RAID array.
	agcount=300	Create 300 allocation groups. This option is set in an attempt to increase performance.

Table 5.3.2.2: XFS File System Creation Options

Create a file system on the logical volume.

```
# mkfs -t xfs -i size=512,maxpct=0 -d su=256k,sw=4,agcount=300
/dev/ROSEsvg/datadomain_lv

meta-data=/dev/ROSEsvg/datadomain_lv isize=512    agcount=300, agsize=894848
blks
        =                               sectsz=512   attr=2, projid32bit=0
data      =                               bsize=4096   blocks=268435456, imaxpct=0
        =                               sunit=64    swidth=256 blks
naming     =version 2                     bsize=4096   ascii-ci=0
log        =internal log                  bsize=4096   blocks=131072, version=2
        =                               sectsz=512   sunit=64 blks, lazy-count=1
realtime   =none                          extsz=4096   blocks=0, rtextents=0
```

A mount point for the logical volume must be created. Since data should not be written to the Red Hat Storage locations directly, the mount point is created in a hidden directory. This should help prevent users from inadvertently writing data to the storage.

Create a mount point for the storage.

```
# mkdir -p /.rhs/ROSEsdatadomain
```

¹² https://access.redhat.com/site/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Performance_Tuning_Guide/s-storage-xfs.html



Once the mount point is created, the file system must be mounted. The options in **Table 5.3.2.3: XFS Mount Options** are used to help increase performance of the file system.

Option	Description
inode64	Allow XFS to create inodes anywhere on the file system.
logbsize=256k	Increase the size of the in-memory log buffer to 256k. The default is 32k.
noatime	Do not update access times when files are read.
nodiratime	Do not update directory inode access times.

Table 5.3.2.3: XFS Mount Options

Edit the `/etc/fstab` file and add an entry for the newly created file system and mount point.

`/etc/fstab`:

```
/dev/ROSESvg/datadomain_lv    /.rhs/ROSESdatadomain    xfs
inode64,logbsize=256k,noatime,nodiratime 1 3
```

Mount the file system.

```
# mount /.rhs/ROSESdatadomain
```

Make sure the file system is mounted with the correct options.

```
# mount -v | grep ROSES
/dev/mapper/ROSESvg-datadomain_lv on /.rhs/ROSESdatadomain type xfs
(rw,noatime,nodiratime,inode64,logbsize=256k)
```

The file system is mounted and the mount point is ready for use as a brick for Red Hat Storage. However, in this environment, a directory for use as the brick location is created on the mounted file system. This makes it easier to identify the data on the disk should the file system become unmounted or detached from this system.

Create a directory on the mounted file system.

```
# mkdir /.rhs/ROSESdatadomain/ROSESdatadomain_brick
```



5.3.3 Gluster

After the underlying storage is configured, the **gluster** service is configured. Gluster can be configured by entering the **gluster** command prompt or by specifying the **gluster** commands on the command line.

Most commands used in configuring gluster are executed on a single storage node. Unless otherwise stated, the following commands are executed on the ra-rhs-srv3 server.

A trusted storage pool must be created before a storage volume can be configured. The **gluster** command **peer probe** is used to add a storage server to the pool. The storage servers each contain a 10 GB network adapter connected to a storage network. A separate name server provides resolution for hosts on the storage network. The FQDN of the 10 GB adapter is the short hostname of the host with a *-10g* appended to the name followed by the storage domain name *storage.se.priv*.

```
# gluster

gluster> peer probe ra-rhs-srv4-10g.storage.se.priv

Probe successful
```

Use the **peer status** command to view the storage servers in the trusted storage pool. The following output shows the newly added peer is connected.

```
gluster> peer status

Number of Peers: 1

Hostname: ra-rhs-srv4-10g.storage.se.priv
Port: 24007
Uuid: 33f54950-319f-45fb-ace7-b97625c33525
State: Peer in Cluster (Connected)
```

The volumes used in this environment are in a replicated configuration. This means that the data stored in one brick is mirrored to another brick. The **volume create** command is used to create the volume.

The command takes the name of the volume as its first argument. The second argument is the type of volume to create, this environment uses a volume type of **replica**. The third argument is the number of bricks to use in the volume. The final options are the bricks to use in the volume. Brick names are specified using the server name followed by a colon (:) and path of the brick.

Create a replicated volume called **ROSESdatadomain**.

```
# gluster
gluster> volume create ROSESdatadomain replica 2 ra-rhs-srv3-
10g.storage.se.priv:/.rhs/ROSESdatadomain/ROSESdatadomain_brick ra-rhs-srv4-
10g.storage.se.priv:/.rhs/ROSESdatadomain/ROSESdatadomain_brick

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



The ROSESdatadomain volume is used as a storage domain for the virtual machine files. Adding the volume to the group virt sets tuning parameters on the volume to increase performance when used as storage domain for virtual machines. When setting the volume to the virt group, the volume must not be used to store any other data¹³.

Add the volume to the virt group.

```
gluster> volume set ROSESdatadomain group virt
Set volume successful
```

The volume is used to store virtual machine files in a Red Hat Enterprise Virtualization environment. The environment requires the user and group ids to be set to 36. This is done using the storage.owner-uid and storage.owner-gid parameters.

Set the correct ownership on the volume.

```
gluster> volume set ROSESdatadomain storage.owner-uid 36
Set volume successful

gluster> volume set ROSESdatadomain storage.owner-gid 36
Set volume successful
```

Use the **volume list** command to view the created volumes.

```
gluster> volume list

ROSESdatadomain
```

Use the **volume status** command to view the functional state of the volume.

```
gluster> volume status

Volume ROSESdatadomain is not started
```

The **volume status** command shows that the volume is not started. Issue the **volume start ROSESdatadomain** command start the volume.

```
gluster> volume start ROSESdatadomain

Starting volume ROSESdatadomain has been successful
```

¹³ https://access.redhat.com/site/documentation/en-US/Red_Hat_Storage/2.0/html-single/Quick_Start_Guide/index.html#chap-Quick_Start_Guide-Virtual_Preparation



The **volume info** command can be used to display information about the configured volumes. Specifying a volume name displays information about the specified volume. View the information for the ROSESdatadomain volume.

```
gluster> volume info ROSESdatadomain

Volume Name: ROSESdatadomain
Type: Replicate
Volume ID: 12a76162-9897-4961-b3df-67754bb0810d
Status: Started
Number of Bricks: 1 x 2 = 2
Transport-type: tcp
Bricks:
Brick1: ra-rhs-srv3-
10g.storage.se.priv:/rhs/ROSESdatadomain/ROSESdatadomain_brick
Brick2: ra-rhs-srv4-
10g.storage.se.priv:/rhs/ROSESdatadomain/ROSESdatadomain_brick
Options Reconfigured:
storage.owner-gid: 36
storage.owner-uid: 36
network.remote-dio: on
cluster.eager-lock: enable
performance.stat-prefetch: off
performance.io-cache: off
performance.read-ahead: off
performance.quick-read: off
```

The volume type, status, bricks, and tuning settings are among the information displayed.

Exit the **gluster** shell using the **exit** command.

```
gluster> exit
```



5.3.4 Firewall

Red Hat Storage uses the ports listed in **Table 5.3.4.1: Gluster Ports** for communication between the storage servers and clients.

Port	Protocol	Service
111	UDP, TCP	Portmap
24007	TCP	Gluster Daemon
24009+	TCP	Ports for bricks. Need one for each brick in a volume.

Table 5.3.4.1: Gluster Ports

Use the **gluster volume info** command and the **grep** command to determine the number of bricks on the server.

```
# gluster volume info | grep "Number of Bricks"
```

```
Number of Bricks: 1 x 2 = 2
```

Only two bricks are used on the server. Only ports 24009 and 24010 are needed for the bricks. However, more bricks will be added to the storage server for future projects. Configure ports 24011 to 24016 as well. This provides enough ports for eight bricks.

Create firewall rules for the needed ports. This step is performed on both the ra-rhs-srv3 and ra-rhs-srv4 systems.

```
# iptables --append INPUT --protocol tcp --match state --state NEW --match tcp --dport 111 --jump ACCEPT
```

```
# iptables --append INPUT --protocol udp --match state --state NEW --match udp --dport 111 --jump ACCEPT
```

```
# iptables --append INPUT --protocol tcp --match state --state NEW --match tcp --dport 24007 --jump ACCEPT
```

```
# iptables --append INPUT --protocol tcp --match state --state NEW --match tcp --dport 24009:24016 --jump ACCEPT
```

```
# service iptables save
```

```
iptables: Saving firewall rules to /etc/sysconfig/iptables:[ OK ]
```



5.3.5 Tuning

The **tuned** daemon monitors the system and dynamically tunes the system settings based upon a selected profile. The **tuned** daemon on the Red Hat Storage server contains profiles to tune the system.

The **tuned-adm** command is used to control the profile currently used by the system. The **list** option is used to display all the available profiles. View the available tuned profiles.

```
# tuned-adm list

Available profiles:
- rhs-high-throughput
- latency-performance
- laptop-ac-powersave
- enterprise-storage
- rhs-virtualization
- desktop-powersave
- spindown-disk
- throughput-performance
- server-powersave
- laptop-battery-powersave
- default
Current active profile: default
```

The **rhs-virtualization** profile tunes the Red Hat Storage server to host virtual machines. Since most of the connections to the storage servers are for virtual machines, the **rhs-virtualization** profile should be used. The profile used by the system is changed using the profile command. Change the profile to **rhs-virtualization**. This step is performed on both the ra-rhs-srv3 and ra-rhs-srv4 systems.

```
# tuned-adm profile rhs-virtualization

Stopping tuned: [ OK ]
Switching to profile 'rhs-virtualization'
Applying ktune sysctl settings:
/etc/ktune.d/tunedadm.conf: [ OK ]
Calling '/etc/ktune.d/tunedadm.sh start': setting readahead to 4096 on brick
devices: dm-3 [ OK ]
Applying sysctl settings from /etc/sysctl.conf
Applying deadline elevator: dm-0 dm-1 dm-2 dm-3 sda sdb sdc [ OK ]
Starting tuned: [ OK ]
```

The **active** command option is used to display the active profile and the current state of the daemon.

```
# tuned-adm active

Current active profile: rhs-virtualization
Service tuned: enabled, running
Service ktune: enabled, running
```



6 Red Hat Enterprise Virtualization Servers

6.1 Red Hat Enterprise Virtualization Hypervisor

The Red Hat Enterprise Virtualization hypervisor is available from the Red Hat Customer Portal¹⁴. Log in to the Red Hat Customer Portal and select **Downloads** at the top of the screen. A page appears listing several technology categories. Select **Download Software** under the **Red Hat Enterprise Linux** category.

A list of software channels appear. Select the plus (+) in front of **Red Hat Enterprise Linux Server (v.6 for 64-bit x86_64)** to expand the list and view the child channels.

Select **Red Hat Enterprise Virtualization Hypervisor (v.6 x86_64)**. An **ISO Image Downloads** page appears. Select the **RHEV-H Image** link. The hypervisor ISO image begins downloading.

Install the server to local storage using the downloaded image using the default configuration options¹⁵.

6.1.1 Configuration

After the Red Hat Enterprise Virtualization Hypervisor is installed and reboots, log into the console as the *admin* user using the password supplied during installation.

The left side of the screen lists different configuration groupings. Use the *ARROW KEYS* to select the **Network** group. The right side of the screen displays the current network configuration. This configuration should be empty.

Use the *TAB* key to cycle through each configuration option listed in **Table 6.1.1.1: RHEV-H Network Configuration** and set each to the appropriate value. Once the values are set, select **Apply** to save the settings.

Option	Value
Hostname	roses-rhevh.example.org
DNS Server 1	10.16.136.36
DNS Server 2	Empty
NTP Server 1	clock.bos.redhat.com
NTP Server 2	Empty

Table 6.1.1.1: RHEV-H Network Configuration

¹⁴ <http://access.redhat.com>

¹⁵ https://access.redhat.com/site/documentation/en-US/Red_Hat_Enterprise_Virtualization/3.3-Beta/html/Hypervisor_Deployment_Guide/index.html



After the settings are applied, the screen returns to the **Network** screen. Select **eth0** and press enter. A configuration screen for eth0 appears. Select the box next to **Static** under the **IPv4 Settings** section. Set the interface configuration according to **Table 6.1.1.2: RHEV-H Nic Configuration**. Select **Apply** to apply the network settings. A **Confirm Network Settings** screen appears. Select **Ok** to configure the networking.

Option	Value
IP Address	10.16.136.78
Netmask	255.255.248.0
Gateway	10.16.143.254

Table 6.1.1.2: RHEV-H Nic Configuration

After the network is configured, the main screen appears again, use the *ARROW KEYS* to select **Security**. Press the *TAB* key until the box to the left of **Enable ssh password authentication** is selected. Press the *SPACEBAR* to place an **asterisk** in the box. Press *TAB* until **Apply** is selected. Press the *SPACEBAR* to apply the settings and start the ssh server.

After the configuration is done, log out of the console by selecting **Status** on the left side of the screen. Use the *TAB* key to select **Log Off**. Press the *SPACEBAR* and the system returns to a login screen.

6.2 Red Hat Enterprise Linux Hypervisor

6.2.1 Network Configuration

Install a second system with the latest version of Red Hat Enterprise Linux 6. Configure the networking as listed in **Table 6.2.1.1: Red Hat Enterprise Linux Hypervisor Networking**.

Option	Value
Hostname	roses-rhelh.example.org
IP Address	10.16.136.79
Network Mask	255.255.248.0
Gateway	10.16.143.254
DNS Server	10.16.136.36

Table 6.2.1.1: Red Hat Enterprise Linux Hypervisor Networking



6.2.2 Updates

After the system is installed, register the system to receive updates. Add the appropriate entitlements or channels as listed in **Table 6.2.2.1: RHEL Hypervisor Entitlements**. These are the entitlements and channels that contain the packages needed to use the system as a hypervisor in a Red Hat Enterprise Virtualization environment. Packages from these entitlements are installed when the system is added to the Red Hat Enterprise Virtualization Manager. As of the writing of this reference architecture, the channels are not available in Red Hat Subscription Manager.

Channel Description	RHN Classic Channel
Red Hat Enterprise Linux 6 Server	rhel-x86_64-server-6
Red Hat Enterprise Virtualization Management Agent	rhel-x86_64-rhev-mgmt-agent-6
Red Hat Storage Native Client	rhel-x86_64-server-rhscclient-6

Table 6.2.2.1: RHEL Hypervisor Entitlements

After the system is installed and subscribed to the appropriate channels, ensure the packages on the system are at the latest available version.

```
# yum -y update  
  
[ ... Output Abbreviated ... ]
```

6.2.3 Time Synchronization

Enable the **ntpd** service and start the service. Check the date and time of the system and make sure it is correct. Fix the date and time if necessary.

```
# chkconfig ntpd on  
  
# service ntpd restart  
  
Starting ntpd: [ OK ]
```

6.2.4 Storage Client

Install the **glusterfs-fuse** package. This package allows the system to communicate with the Red Hat Storage Server.

```
# yum -y install glusterfs-fuse  
  
[ ... Output Abbreviated ... ]
```



6.2.5 SSH Configuration

SSH must be configured to allow root to log into the system remotely. Edit the `/etc/ssh/sshd_config` file and make sure the **PermitRootLogin** parameter is set to yes.

```
PermitRootLogin yes
```

Restart the **ssh** daemon.

```
# service sshd restart
Stopping sshd:
Starting sshd:
```

```
[ OK ]
[ OK ]
```

6.3 Red Hat Enterprise Virtualization Manager

The Red Hat Enterprise Virtualization Manager runs as a service on a Red Hat Enterprise Linux system. Install a system with the latest version of Red Hat Enterprise Linux 6.

6.3.1 Network Configuration

Configure the networking as listed in **Table 6.3.1.1: roses-rhevmm Network Configuration**.

Setting	Value
Hostname	roses-rhevmm.example.org
IP Address	10.16.136.31
Network Mask	255.255.248.0
Gateway	10.16.143.254
DNS Server	10.16.136.36

Table 6.3.1.1: roses-rhevmm Network Configuration



6.3.2 Updates

After the system is installed, register the system to receive updates. Add the appropriate entitlements or channels as listed in **Table 6.3.2.1: RHEVM Entitlements**.

Channel Description	Subscription Manager	RHN Classic Channels
Red Hat Enterprise Linux 6 Server	Red Hat Enterprise Linux Server	rhel-x86_64-server-6
Red Hat Enterprise Linux 6 Server Supplementary	Not Needed	rhel-x86_64-server-supplementary-6
Red Hat Enterprise Virtualization Manager	Red Hat Enterprise Virtualization	rhel-x86_64-server-6-rhev-3.2
Red Hat JBoss Enterprise Application Platform	JBoss Enterprise Application Platform	jbappplatform-6-x86_64-server-6-rpm

Table 6.3.2.1: RHEVM Entitlements

After the system is installed and subscribed to the appropriate channels, ensure the packages on the system are at the latest available version.

```
# yum -y update  
[ ... Output Abbreviated ... ]
```

6.3.3 Time Synchronization

Enable the **ntpd** service and start the service. Check the date and time of the system and make sure it is correct. Fix the date and time if necessary.

```
# chkconfig ntpd on  
# service ntpd restart  
Starting ntpd: [ OK ]
```



6.3.4 Red Hat Enterprise Virtualization Manager Configuration

Install the Red Hat Enterprise Virtualization Manager software by installing the **rhev**m package. Installing this package also installs its dependency packages.

```
# yum -y install rhvm  
[ ... Output Abbreviated ... ]
```

Install the documentation for Red Hat Enterprise Virtualization Manager. This is not necessary, but may be useful in the future.

```
# yum -y install rhvm-doc  
[ ... Output Abbreviated ... ]
```

The Red Hat Enterprise Virtualization Manager software is configured using the **rhev**m-**setup** command. This command configures the certificates, database, firewall, web server, and other items required for a correctly functioning Red Hat Enterprise Virtualization Manager server.

The **rhev**m-**setup** command prompts the user for various configuration during the process. This behavior can be changed by supplying the **--answer-file** option. This option points to a file that contains predetermined configuration for the **rhev**m-**setup** command to use.

Create an answer file called *rhvm-answer.txt* that contains the following:

```
[general]  
OVERRIDE_HTTPD_CONFIG=yes  
HTTP_PORT=80  
HTTPS_PORT=443  
RANDOM_PASSWORDS=no  
MAC_RANGE=00:1A:4A:A8:6E:00-00:1A:4A:A8:6E:FF  
HOST_FQDN=roses-rhvm.example.org  
AUTH_PASS=[Password]  
ORG_NAME=example.org  
DC_TYPE=ISCSI  
DB_REMOTE_INSTALL=local  
DB_HOST=  
DB_PORT=5432  
DB_ADMIN=postgres  
DB_REMOTE_PASS=  
DB_SECURE_CONNECTION=no  
DB_LOCAL_PASS=[Password]  
NFS_MP=/var/lib/exports/iso  
ISO_DOMAIN_NAME=ISO_DOMAIN  
CONFIG_NFS=no  
OVERRIDE_FIREWALL=iptables
```



Configure the Red Hat Enterprise Virtualization Manager using the answer file.

```
# rhvm-setup --answer-file rhvm-answer.txt

Welcome to RHEV Manager setup utility

Installing:
Configuring RHEV Manager...      [ DONE ]
Configuring JVM...               [ DONE ]
Creating CA...                   [ DONE ]
Updating ovirt-engine service... [ DONE ]
Setting Database Configuration... [ DONE ]
Setting Database Security...     [ DONE ]
Creating Database...             [ DONE ]
Updating the Default Data Center Storage Type... [ DONE ]
Editing RHEV Manager Configuration... [ DONE ]
Editing Postgresql Configuration... [ DONE ]
Configuring Firewall...         [ DONE ]
Starting ovirt-engine Service... [ DONE ]
Configuring HTTPD...            [ DONE ]

**** Installation completed successfully ****

(Please allow RHEV Manager a few moments to start up.....)

**** To access RHEV Manager browse to http://roses-rhvm.example.org:80 ****

[ ... Output Abbreviated ... ]
```

The Red Hat Enterprise Virtualization Manager administrative portal is accessed using the URL provided in the output. Open a browser and enter <http://roses-rhvm.example.org:80> for the url. The Red Hat Enterprise Virtualization Manager portal should appear.



7 Red Hat Enterprise Virtualization Environment

To configure the Red Hat Enterprise Virtualization environment, login to the Red Hat Enterprise Virtualization Administration Portal by opening a supported browser and entering <https://roses-rhevm.example.org/webadmin> as the url. Use **admin** as the user name and supply the password entered in the answer file that was used during the installation of the Red Hat Enterprise Virtualization Manager packages.

7.1 Data Center Configuration

Under the **Data Centers** tab, select **New**.

A **New Data Center** screen appears.

Create a new data center. Enter **DC-Main** for the Name. The Red Hat Storage Servers provide a POSIX compliant file system, set the **Type** of the data center to **POSIX compliant FS**. Ensure **Quota Mode** is set to **Disabled**. Select **OK**.

A **New Data Center - Guide Me** window appears. Select **Configure Later**.

7.2 Cluster Configuration

Under the **Clusters** tab, select **New**.

A **New Cluster** screen appears.

Create a new cluster. Select **DC-Main** for the data center. Enter **CL-Production** for the Name. Set the **CPU Name** to **Intel Nehalem Family**. Ensure the **Enable Virt Service** option is selected. Do not select **Enable Gluster Service**. Select **OK**.

A **New Cluster - Guide Me** window appears. Select **Configure Later**.



7.3 Hypervisor Configuration

7.3.1 Red Hat Enterprise Virtualization Hypervisor

Login to the console of the `roses-rhevh.example.org` hypervisor. Using the *ARROW KEYS*, select **RHEV-M** in the right column. The following screen appears.

```
RHEV Hypervisor 6.4-20130528.0.el6_4
roses-rhevm.example.org
```

Status	RHEV-M Configuration
Network	Management Server: roses-rhevm.example.org
Security	Management Server Port: 443__
Keyboard	
SNMP	
Logging	
Kernel Dump	[*] Connect to RHEV-M and Validate Certificate
Remote Storage	
CIM	Optional password for adding node through RHEV-M UI
RHEV-M	Password: *****
Plugins	Confirm Password: *****
Red Hat Network	

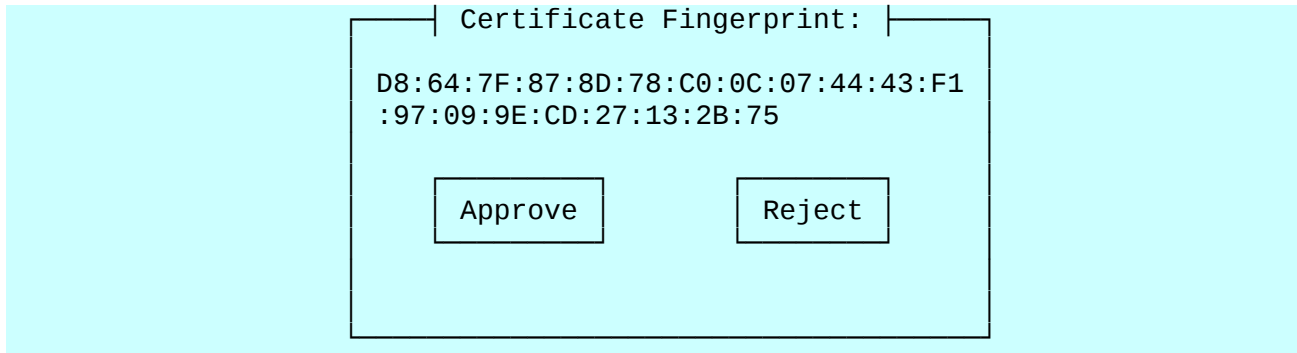
<Apply> <Reset>

Use the *TAB* key to move around the selections. Enter ***roses-rhevm.example.org*** for the **RHEV-M Configuration Management Server**. Enter **443** for the **Management Server Port**. Ensure the box next to **Connect to RHEV-M and Validate Certificate** contain an asterisk. Enter the root password for the RHEVM sever under **Password** and **Confirm Password**.

Use the *TAB* key to select **Apply** and press the *SPACEBAR* to apply the changes.



A window appears prompting the approval of the rhev managers certificate. Select **Approve**.



The configuration of the Red Hat Enterprise Virtualization hypervisor is complete.

On the Red Hat Enterprise Virtualization Manager portal, select the **Hosts** tab. Select the Red Hat Enterprise Virtualization hypervisor host. This host is currently in a status of *PENDING APPROVAL*.

Select **Approve**.

An **Edit and Approve Host** screen appears. Select **DC-Main** for the **Data Center** and **CL-Production** for the **Host Cluster**. Select **OK**.

A power management screen appears. Select **OK** to configure power management later.

The **Status** of the host changes to **Installing** and then to **Up**.

7.3.2 Red Hat Enterprise Linux Hypervisor

From the Red Hat Enterprise Virtualization Manager portal, select the **Hosts** tab. Select **New**.

A **New Host** screen appears. Enter **roses-rhelh.example.org** as the **Name** of the Red Hat Enterprise Linux Hypervisor. Enter an IP address of **10.16.136.79** in the **Address** field. Complete the **Root Password** field with the correct password. Select **Automatically configure host firewall**. Select **OK**.

A message appears indication that power management is not configured. Select **OK** to configure power management later.

The **Status** of the host changes to **Installing**, then to **Reboot** and finally **Up**.



7.3.3 Power Management

Select the **Hosts** tab in the Red Hat Enterprise Virtualization Manager portal. Click the right mouse button on the **roses-rhelh.example.org** host and select **Edit**.

An **Edit Host** window appears, select **Power Management** on the left side of the window.

Select **Enable Power Management**. Set power management to **Primary**. Deselect **Concurrent**. Enter **10.16.143.238** for the **Address**, this is the IP address of the Integrated Lights Out interface for the physical host. Enter **Administrator** for the **User Name**. Enter the password for the **Password**. Select **ilo2** for the **Type**. Leave the remaining options set to their defaults.

Scroll down in the window if needed and look for a **Test** button. Select this button.

A status of “*TESTING IN PROGRESS. IT WILL TAKE A FEW SECONDS. PLEASE WAIT...*” appears below the button. After a few minutes “*TEST SUCCEEDED, ON*” appears.

Select **OK** to accept the power management configuration.

Repeat the steps for the **roses-rhev.example.org** host. Use 10.16.143.235 for the Address.



7.4 Network Configuration

Under the **Networks** tab, select **New**.

A **New Logical Network** screen appears. Choose **DC-Main**, for the **Data Center**. Enter **RHStorage** for the **Name**. Ensure **VM Network** is selected. Select **Attach** for the CL-Production cluster. Ensure **Allow all users to use this Network** is selected. Select **OK**.

Each host in the cluster must be configured to use the **RHStorage** network. Select the **roses-rhelh.example.org** host under the **Hosts** tab. Select the **Network Interfaces** tab at the bottom of the screen. Select **Setup Host Networks**. A screen similar to **Figure 7.4.1**: appears.

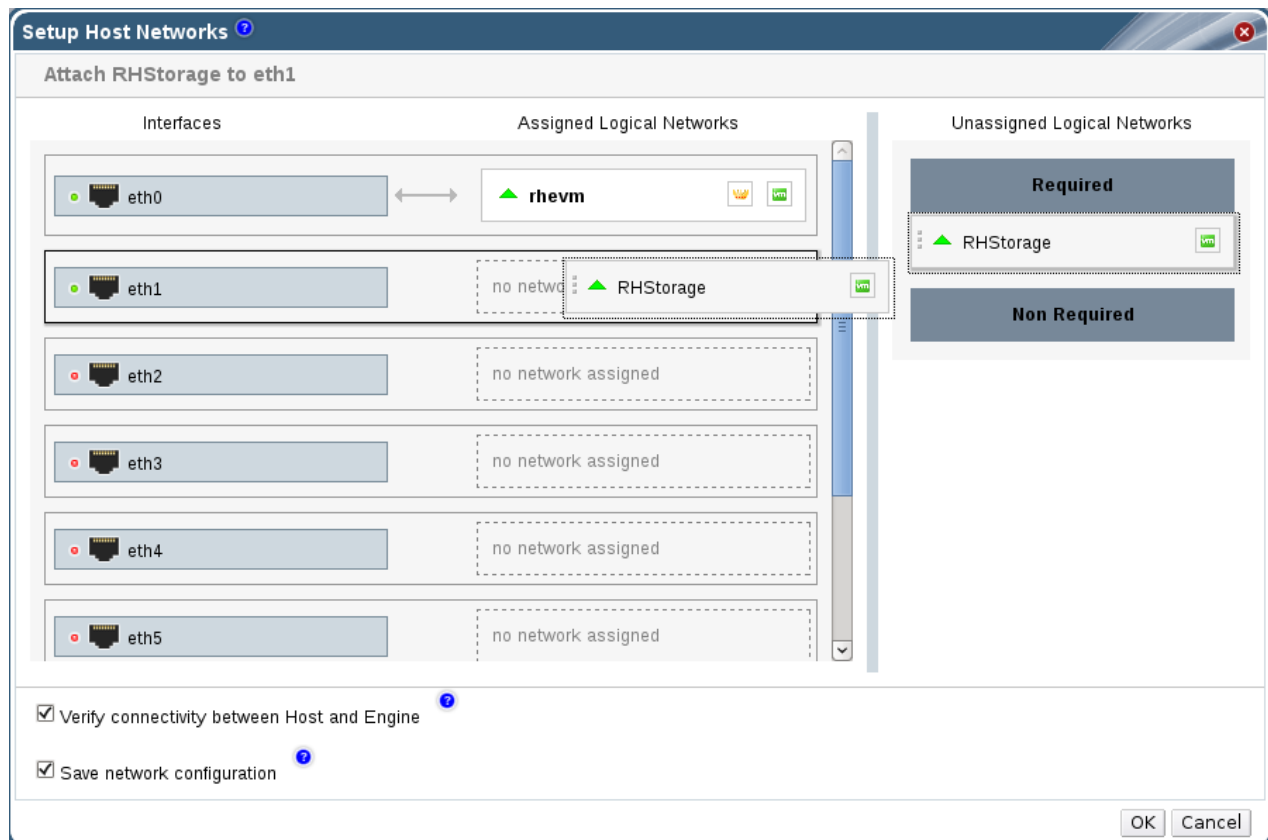


Figure 7.4.1:

Select and hold the mouse button on the **RHStorage** network under the **Unassigned Logical Networks**. While holding the mouse button down, move the **RHStorage** network over the **eth1** interface. The network should appear under the **Assigned Logical Networks** column next to the **eth1** interface.



Move the mouse over the **RHStorage** network, a small pencil shaped icon appears. Select the pencil icon.

An **Edit Network RHStorage** window appears. Select **Static** under **Boot Protocol**. Enter **172.31.136.79** for the **IP** and **255.255.0.0** for the **Subnet Mask**. Select **OK**.

The **Edit Network RHStorage** window closes. On the **Setup Hosts Network** window, select **Save network configuration**. Select **OK**.

After a few moments, the **eth1** interface changes to an operational state.

Repeat these steps to configure the RHStorage network on the roses-rhev.example.org system. Use 172.31.136.78 for the IP address.

7.5 Storage Domain Configuration

Select the **Storage** tab. Select **New Domain**.

A **New Domain** window opens. Enter **Data-DC-Main** as the **Name**. Choose the **DC-Main Data Center**. Choose **Data / POSIX compliant FS** for **Domain Function / Storage Type**. Select **roses-rhelh.example.org** for **Use Host**. Enter **ra-rhs-srv3-10g.storage.se.priv:/ROSESdatadomain** for the **Path**. Enter **glusterfs** for the **VFS Type**. Select **OK**.

After a few moments the **Data-DC-Main** domain appears under the storage tab.

Selecting the **DC-Main** data center under the **Data Centers** tab indicates the **DC-Main** data center has a **Status** of **Locked**. After a few moments, the **Data Center Status** changes to **Active**.

The **Storage** tab indicates the **Data-DC-Main** storage domain has a **Cross Data-Center Status** of **Active**.



8 OpenShift Enterprise

This section configures a simple Red Hat OpenShift Enterprise environment consisting of a single broker host and two node hosts. The hosts used are virtual machines within the Red Hat Enterprise Virtualization environment. The second node host is used as a failover host for the first node host.

1. Install the broker host.
2. Install a single node host.
3. Create application gears.
4. Prepare the node host for high-availability.
5. Install a second node host for use as a backup node host.
6. Fail the first node host over to the second node host.

8.1 Broker Host

Create a virtual machine to act as a broker by selecting **New Server** under the **Virtual Machines** tab in the Red Hat Enterprise Virtualization Manager portal.

Use a name of roses-broker and the information in **Table 3.2: OSE Virtual Machine Configuration** to create the virtual machine. Perform a basic installation of Red Hat Enterprise Linux 6.4.

8.1.1 Network Configuration

Configure the host networking as described in **Table 8.1.1.1: Broker Host Networking**.

Setting	Value
Hostname	roses-broker.example.org
IP Address	10.16.136.76
Network Mask	255.255.248.0
Gateway	10.16.143.254
Name Resolution	10.16.136.36

Table 8.1.1.1: Broker Host Networking



8.1.2 Updates

Register the virtual machine to receive updates. The broker host must be subscribed to the appropriate channels to install the broker software. Using **Table 8.1.2.1: Broker Subscriptions** as a reference, attach the broker host to the correct channel subscriptions.

Channel	Subscription Manager	RHN Classic
Broker Infrastructure	OpenShift Enterprise Broker Infrastructure	rhel-x86_64-server-6-ose-1.2-infrastructure
Client Tools	OpenShift Enterprise	rhel-x86_64-server-6-ose-1.2-rhc
Red Hat Enterprise Virtualization Guest Agent	Red Hat Enterprise Linux Server	rhel-x86_64-rhev-agent-6-server
Software Collection		rhel-x86_64-server-6-rhsc1-1

Table 8.1.2.1: Broker Subscriptions

Install the Red Hat Enterprise Virtualization guest agent for the virtual machine.

```
# yum -y install rhvm-guest-agent
# service ovirt-guest-agent start
# chkconfig ovirt-guest-agent on
```

Packages from third-party repositories or other Red Hat products can conflict with the packages installed for OpenShift Enterprise. Adjusting the priorities of the repositories and excluding certain package updates can prevent this.

Scripts are available at <https://github.com/openshift/openshift-extras/tree/enterprise-2.0/admin> that check for missing update channels and incorrect priorities on yum repositories. The names of the scripts at this location can change. At the time of writing this reference architecture, the following files were available.



Download the scripts from <https://github.com/openshift/openshift-extras/tree/enterprise-2.0/admin>. The scripts needed are under the *yum-validator* folder. The files under the *yum-validator/etc* folder are also placed in the */root* directory. The files under the *yum-validator/yumvalidator* directory are placed in the */root/yumvalidator* directory.

```
# wget \  
> https://github.com/openshift/openshift-extras/raw/enterprise-  
2.0/admin/yum-validator/oo-admin-yum-validator \  
> https://github.com/openshift/openshift-extras/raw/enterprise-  
2.0/admin/yum-validator/etc/repos.ini \  
> https://github.com/openshift/openshift-extras/raw/enterprise-  
2.0/admin/yum-validator/etc/beta2.ini  
  
# mkdir yumvalidator  
# cd yumvalidator  
# wget \  
> https://github.com/openshift/openshift-extras/raw/enterprise-  
2.0/admin/yum-validator/yumvalidator/check_sources.py \  
> https://github.com/openshift/openshift-extras/raw/enterprise-  
2.0/admin/yum-validator/yumvalidator/repo_db.py \  
> https://github.com/openshift/openshift-extras/raw/enterprise-  
2.0/admin/yum-validator/yumvalidator/__init__.py  
  
# cd ..
```

After the files are downloaded, execute the **oo-admin-yum-validator** script, specify the command line option **--report-all**. This option displays all the issues the script finds. Also specify the **--role broker** and **--role client** options. These specify the OpenShift Enterprise roles of the host. Specify the **--repo-config repos.ini** to tell the script to use the *repos.ini* file for the repository data.



The script indicates the **yum-plugin-priorities** package is not installed and a few other errors.

```
# ./oo-admin-check-sources.py --report-all --role broker --role client  
--repo-config repos.ini
```

```
Detected OpenShift Enterprise repository subscription managed by RHN Classic  
or RHN Satellite.
```

```
Detected installed OpenShift Enterprise version 1.2
```

```
Checking if yum-plugin-priorities is installed
```

```
Required package yum-plugin-priorities is not installed. Install the package  
with the following command:
```

```
# yum install yum-plugin-priorities
```

```
Checking channel/repository priorities
```

```
Resolving repository/channel/subscription priority conflicts
```

```
To resolve conflicting repositories, update
```

```
/etc/yum/pluginconf.d/rhnplugin.conf with the following changes:
```

```
    Set priority=10 in the [rhel-x86_64-server-6-rhsc1-1] section
```

```
    Set priority=10 in the [rhel-x86_64-server-6-ose-1.2-rhc] section
```

```
    Set priority=10 in the [rhel-x86_64-server-6-ose-1.2-infrastructure]
```

```
section
```

```
    Set priority=20 in the [rhel-x86_64-server-6] section
```

```
The following repositories need package exclusions set:
```

```
    rhel-x86_64-server-6
```

```
Make the following modifications to /etc/yum/pluginconf.d/rhnplugin.conf
```

```
Add the following line to the [rhel-x86_64-server-6] section:
```

```
    exclude=tomcat6*
```

```
Please re-run this tool after making any recommended repairs to this system
```

The missing package must be installed manually. Install the **yum-plugin-priorities** package.

```
# yum -y install yum-plugin-priorities
```




Execute the command using the **--fix-all** option. This option instructs the script to attempt to repair any issues it encounters.

```
# ./oo-admin-check-sources.py --fix-all --role broker --role client --repo-config repos.ini

No roles have been specified. Attempting to guess the roles for this system...
If the roles listed below are incorrect or incomplete, please re-run this script with the appropriate --role arguments
    broker
    client
Detected OpenShift Enterprise repository subscription managed by RHN Classic or RHN Satellite.
Detected installed OpenShift Enterprise version 1.2

Checking if yum-plugin-priorities is installed
Checking channel/repository priorities
Resolving repository/channel/subscription priority conflicts
Setting priority for repository rhel-x86_64-server-6-rhsc1-1 to 10
Setting priority for repository rhel-x86_64-server-6-ose-1.2-rhc to 10
Setting priority for repository rhel-x86_64-server-6-ose-1.2-infrastructure to 10
Setting priority for repository rhel-x86_64-server-6 to 20
```

Execute the script again to identify any more issues with the repositories. No issues are found.

```
# ./oo-admin-check-sources.py --report-all --role broker --role client --repo-config repos.ini

No roles have been specified. Attempting to guess the roles for this system...
If the roles listed below are incorrect or incomplete, please re-run this script with the appropriate --role arguments
    broker
    client
Detected OpenShift Enterprise repository subscription managed by RHN Classic or RHN Satellite.
Detected installed OpenShift Enterprise version 1.2

Checking if yum-plugin-priorities is installed
Checking channel/repository priorities
No problems could be detected!
```

Update the packages on the system to make sure the latest are being used.

```
# yum -y update
```



8.1.3 Time Synchronization

Enable and start the **ntpd** service to ensure the system has the correct time.

```
# chkconfig ntpd on
```

```
# service ntpd start
```

```
Starting ntpd: [ OK ]
```

8.1.4 MongoDB

The broker uses a Mongo database to store information about the users and OpenShift Enterprise Environment.

Install the **mongodb-server** package.

```
# yum -y install mongodb-server
```

The Mongo database must be configured to require authentication. Edit the */etc/mongodb.conf* file and make sure the **auth** line is set to **true**.

```
auth = true
```

Configure the Mongo database to use a small database file by default. Make sure the configuration file contains a definition for **smallfiles** and that it is set to **true**.

```
smallfiles = true
```

Start the Mongo database and ensure it starts upon boot.

```
# service mongod start
```

```
Starting mongod: [ OK ]
```

```
# chkconfig mongod on
```



Create an administrative account for the Mongo database. This allows management of the databases.

```
# mongo

MongoDB shell version: 2.2.3
connecting to: test

> use admin

switched to db admin

> db.addUser("admin", "[Password]")

{
  "user" : "admin",
  "readOnly" : false,
  "pwd" : "qwertyuiop1234567890",
  "_id" : ObjectId("5227966f696d7d1aae36d002")
}
addUser succeeded, but cannot wait for replication since we no longer have
auth

> db.auth("admin", "[Password]")

1
```

Create a user account within the **openshift_broker** database called **openshift**. This account is used by the broker to access the database.

```
> use openshift_broker

switched to db openshift_broker

> db.addUser("openshift", "[Password]")

{
  "user" : "openshift",
  "readOnly" : false,
  "pwd" : "1234567890qwertyuiop",
  "_id" : ObjectId("522796ad696d7d1aae36d003")
}

> [CTRL-D]

bye
```



Verify that the **openshift** user account works.

```
# mongo

MongoDB shell version: 2.2.3
connecting to: test

> use openshift_broker
switched to db openshift_broker

> db.auth("openshift", "[Password]")
1

> db.system.users.find()
{ "_id" : ObjectId("522796ad696d7d1aae36d003"), "user" : "openshift",
"readOnly" : false, "pwd" : "1234567890qwertyuiop" }

> [CTRL-D]
bye
```

8.1.5 ActiveMQ

ActiveMQ is a messaging service. Install ActiveMQ and the ActiveMQ client.

```
# yum -y install activemq activemq-client
```

ActiveMQ is configured using the `/etc/activemq/activemq.xml` file. The configuration file installed with the package may be used, but requires editing to add plugins and authentication to the configuration. Another option is to download the `activemq.xml` configuration file from **github**. This file contains the authentication information and requires less editing than the original file.

Download the `activemq.xml` file from **github**.

```
# cd /root

# wget https://raw.githubusercontent.com/openshift/openshift-extras/enterprise-1.2/enterprise/install-scripts/activemq.xml

--2013-09-11 16:36:25-- https://raw.githubusercontent.com/openshift/openshift-extras/enterprise-1.2/enterprise/install-scripts/activemq.xml
Resolving raw.githubusercontent.com... 199.27.77.133
Connecting to raw.githubusercontent.com|199.27.77.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 6900 (6.7K) [text/plain]
Saving to: "activemq.xml"

100%[=====>] 6,900 --.-K/s in 0s
```



Make a backup copy of the original configuration file.

```
# cp /etc/activemq/activemq.xml{,.orig}
```

Copy the downloaded *activemq.xml* file to */etc/activemq/activemq.xml*.

```
# cp activemq.xml /etc/activemq/activemq.xml
cp: overwrite '/etc/activemq/activemq.xml'? y
```

The *activemq.xml* must be edited to reflect this environment. Change the broker name in the file as well as the default passwords.

```
# sed -i \
> -e 's/\\(brokerName=\\)activemq.example.com/\\1roses-broker.example.org/' \
> -e 's/\\(username="mcollective" password=\\)marionette/\\1[Password]/' \
> -e 's/\\(username="admin" password=\\)secret/\\1[Password]/' \
> /etc/activemq/activemq.xml
```

By default, the ActiveMQ monitor console allows non-authenticating access from any interface. For security reasons, this should be changed to require authentication and to only allow access from the localhost. This configuration is stored in the */etc/activemq/jetty.xml* file.

Authentication is controlled by the *authenticate* property of the bean with the **securityConstraint** id. This property must have a value of **true** to enable authentication. Host access is controlled by the **host** property under the **Server** bean. This property must be added to the configuration under the **Connector** and **SecureConnector** beans and must have a value of **127.0.0.1** to restrict access to the localhost.

Make the changes to the */etc/activemq/jetty.xml* file.

```
# sed -i -e '/name="authenticate"/s/false/true/' /etc/activemq/jetty.xml
# sed -i -e '/name="port"/a<property name="host" value="127.0.0.1" />'
/etc/activemq/jetty.xml
```

The */etc/activemq/jetty-realm.properties* file contains the user, password, and role definitions for accessing ActiveMQ. Entries in the file are in the format of:

user: password, roles

Change the password of the admin user.

```
# grep admin /etc/activemq/jetty-realm.properties
admin: admin, admin

# sed -i -e '/admin:/s/admin,/ [Password],/' /etc/activemq/jetty-
realm.properties

# grep admin /etc/activemq/jetty-realm.properties
admin: [Password], admin
```



Enable the **activemq** service to start on boot and start the service now.

```
# chkconfig activemq on  
  
# service activemq start  
Starting ActiveMQ Broker...
```

The ActiveMQ service listens for requests from Mcollective on TCP port 61613. Open this port on the firewall.

```
# firewall-cmd --add-port=61613/tcp --permanent
```

ActiveMQ runs the web console on TCP port 8161. Verify that authentication with ActiveMQ is working by using the curl command to request the topics url on the localhost. Use the admin user account to connect to the url <http://localhost:8161/admin/xml/topics.jsp>. Specify an incorrect password for the connection. The command returns a **401 Unauthorized** response.

```
# curl --head --user admin:incorrectpassword  
http://localhost:8161/admin/xml/topics.jsp  
  
HTTP/1.1 401 Unauthorized  
WWW-Authenticate: basic realm="ActiveMQRealm"  
Cache-Control: must-revalidate,no-cache,no-store  
Content-Type: text/html;charset=ISO-8859-1  
Content-Length: 1293  
Server: Jetty(7.6.7.v20120910)
```

Issue the command again using the correct password. The command returns a **200 OK** response indicating a successful authentication.

```
# curl --head --user admin:[Password]  
http://localhost:8161/admin/xml/topics.jsp  
  
HTTP/1.1 200 OK  
Set-Cookie: JSESSIONID=mvqfu5zs9grj3y5v1xghm7b7;Path=/admin  
Expires: Thu, 01 Jan 1970 00:00:00 GMT  
Content-Type: text/xml;charset=ISO-8859-1  
Content-Length: 185  
Server: Jetty(7.6.7.v20120910)
```

Verify the ActiveMQ service is returning topic data.

```
# curl --user admin:[Password] --silent  
http://localhost:8161/admin/xml/topics.jsp | grep topic  
  
<topics>  
<topic name="ActiveMQ.Advisory.Queue">  
</topic>  
  
[ ... Output Abbreviated ... ]
```



8.1.6 MCollective

Mcollective is a service that acts like a client to send and receive messages between OpenShift Enterprise and the ActiveMQ messaging service. The broker uses the MCollective client to communicate with the **mcollective** service that is installed on the node hosts.

Install the MCollective client.

```
# yum -y install mcollective-client
```

The configuration file for the MCollective client is named */etc/mcollective/client.cfg*. Replace the current configuration file with the following contents. This configuration configures the MCollective client to use the ActiveMQ service for message passing. A preshared key is also defined. This key must match the key used in the *server.cfg* file on the node hosts.

/etc/mcollective/client.cfg:

```
topicprefix = /topic/
main_collective = mcollective
collectives = mcollective
libdir = /opt/rh/ruby193/root/usr/libexec/mcollective
logfile = /var/log/mcollective-client.log
loglevel = debug
direct_addressing = 1

# Plugins
securityprovider = psk
plugin.psk = [pre-shared key]

connector = activemq
plugin.activemq.pool.size = 1
plugin.activemq.pool.1.host = roses-broker.example.org
plugin.activemq.pool.1.port = 61613
plugin.activemq.pool.1.user = mcollective
plugin.activemq.pool.1.password = [Password]

# Facts
factsource = yaml
plugin.yaml = /etc/mcollective/facts.yaml
```



8.1.7 Broker application

The broker application provides interfaces to communicate and interact with the OpenShift Enterprise platform. These interfaces include a web interface and a REST API. Install the OpenShift Enterprise broker and applications.

```
# yum -y install openshift-origin-broker openshift-origin-broker-util  
rubygem-openshift-origin-auth-remote-user rubygem-openshift-origin-msg-  
broker-mcollective rubygem-openshift-origin-dns-nsupdate
```

The default `mod_ssl` installation creates a virtual host that can interfere and cause issues within the OpenShift Enterprise broker application. Remove the **VirtualHost** definition from the `/etc/httpd/conf.d/ssl.conf` configuration file.

```
# cp /etc/httpd/conf.d/ssl.conf{,.orig}  
  
# sed -i '/VirtualHost/,/VirtualHost/ d' /etc/httpd/conf.d/ssl.conf
```

The broker application places Apache configuration files in `/etc/httpd/conf.d`. The `000002_openshift_origin_broker_servername.conf` file contains the hostname of the server. The default hostname must be changed to reflect the servers FQDN. Set the correct hostname in the brokers Apache configuration file.

```
# sed -i -e "s/ServerName .*/ServerName `hostname`/"  
/etc/httpd/conf.d/000002_openshift_origin_broker_servername.conf
```

The broker application can log information to a log file. This log file must be created and owned by the **apache** user for the system to use it. Create the file and set the appropriate permissions.

```
# touch /var/log/mcollective-client.log  
  
# chown -v apache:root /var/log/mcollective-client.log
```

Start the Apache web service (**httpd**) and make sure it is set to start on boot.

```
# service httpd start  
  
# chkconfig httpd on
```

The firewall must allow incoming connections to the Apache service. Open the correct ports in the firewall.

```
# lokkit --service=https  
  
# lokkit --service=http
```




The broker application uses access keys to authenticate various OpenShift Enterprise services. The **openssl** command is used to create the key pair for the authentication. First a 2048 bit private key is created. A public key is then created using the private key.

Use the **openssl** command to create the private and public keys.

```
# openssl genrsa -out /etc/openshift/server_priv.pem 2048

Generating RSA private key, 2048 bit long modulus
.....+++
.....+++
e is 65537 (0x10001)

# openssl rsa -in /etc/openshift/server_priv.pem -pubout >
/etc/openshift/server_pub.pem

writing RSA key
```

At times, the node must contact the broker for various tasks. Some tasks, such as the creation and destruction of gears, require the action to be performed by a user or on behalf of a user. To do this securely, OpenShift Enterprise uses an encrypted authentication token. This token is unique and specific to each gear. A salted token is created using the value of **AUTH_SALT** in the */etc/openshift/broker.conf* file. A salt value can be easily created using the **openssl** command to produce a base64 encoded random string.

Create a new salt string.

```
# openssl rand -base64 64

xuZvaiVS9Ba7FmbiIHQdwCQEF86VceaB8S+cos+c/bxNufpQCCQmNhAz2h3T8hbs
gYOLV+KEeRcUJZXLfvWSww==
```

Make a backup copy of the */etc/openshift/broker.conf* file.

```
# cp /etc/openshift/broker.conf{,.orig}
```

Change the **AUTH_SALT** value in the */etc/openshift/broker.conf* file to contain the newly created salt string.

```
# sed -i
'/AUTH_SALT/cAUTH_SALT="xuZvaiVS9Ba7FmbiIHQdwCQEF86VceaB8S+cos+c/bxNufpQCCQm
NhAz2h3T8hbsgYOLV+KEeRcUJZXLfvWSww=="' /etc/openshift/broker.conf
```



Brokers use a salted value to validate that incoming requests have not been hijacked. The value of **SESSION_SECRET** from the `/etc/openshift/broker.conf` file is used for this salt value. This salt value is created using the **openssl** command, but is hex encoded instead of base64 encoded.

Create a new salt value for broker to broker communication.

```
# openssl rand -hex 64

e475618b3b2cb05113996cd2a4631889212f406961d9d57f7ccd5f5d5fb222932f408b2eb2105
ca38bbfa0d4d7a8e66ae43b83b147b1c534e8b4afdf02e89e469
```

Change the value of **SESSION_SECRET** to reflect the newly created salt value.

```
# sed -i
'/SESSION_SECRET/cSESSION_SECRET="e475618b3b2cb05113996cd2a4631889212f406961
d9d57f7ccd5f5d5fb222932f408b2eb2105ca38bbfa0d4d7a8e66ae43b83b147b1c534e8b4afd
f02e89e469"' /etc/openshift/broker.conf
```

At times, such as the moving of gears between nodes, the brokers must communicate to the nodes using the SSH protocol. To do this, a key pair must be generated for **ssh** to use for passwordless authentication. The private key is kept on the broker, while the public key is placed on the nodes.

Generate a key pair using the **ssh-keygen** command.

```
# ssh-keygen -t rsa -b 2048 -f ~/.ssh/rsync_id_rsa

Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase): [Press Enter]
Enter same passphrase again: [Press Enter]
Your identification has been saved in /root/.ssh/rsync_id_rsa.
Your public key has been saved in /root/.ssh/rsync_id_rsa.pub.
The key fingerprint is:
59:e4:09:f2:5f:f4:6a:b9:fb:c2:11:cf:8b:81:a7:8a root@roses-
broker.cloud.lab.eng.bos.redhat.com
The key's randomart image is:
+--[ RSA 2048 ]-----+
|      . . . . .      |
|      o + o .       |
|      . + . .       |
|      + ..o         |
|      S ..++        |
|      ..+.o         |
|      ++. +         |
|      . . +..       |
|      E .. .o.      |
+-----+

# cp ~/.ssh/rsync_id_rsa* /etc/openshift/
```



Several SELinux boolean variables must be enabled for the broker application to function correctly. The booleans are listed in **Table 8.1.7.1: SELinux Booleans for the Broker Application**.

Boolean	Description
httpd_unified	Allows httpd processes and scripts to read, write, and execute appropriately labeled content.
httpd_execmem	Allows httpd scripts and modules execmem.
httpd_can_network_connect	Allows httpd scripts and modules can connect to the network using any port.
httpd_can_network_relay	Allows httpd to act as a relay.
httpd_run_stickshift	Allows httpd to run in stickshift mode and not transition to passenger.
named_write_master_zones	Allow the named service to write the master zone files. This is required for dynamic dns updates.
allow_ybind	Allows broker system to use NIS.

Table 8.1.7.1: SELinux Booleans for the Broker Application

Enable the SELinux booleans in **Table 8.1.7.1: SELinux Booleans for the Broker Application**.

```
# setsebool -P httpd_unified=on httpd_execmem=on  
httpd_can_network_connect=on httpd_can_network_relay=on  
httpd_run_stickshift=on named_write_master_zones=on allow_ybind=on
```

Since the SELinux booleans have been changed, some files and directories need the contexts fixed on them. The files that need to be corrected were installed by the **ruby193-rubygem-passenger** and **ruby193-mod_passenger** packages as well as the directories and files in the `/var/run` and `/opt` directories.

The **fixfiles** command uses the **-R** option to query the rpm database for a list of files belonging to a package. This command is used to restore the contexts of files belonging to the packages.



The **restorecon** command uses the **-r** and **-v** option to restore the contexts of all the files and directories in the **/var/run** and **/opt** directories. The **-r** option instructs the **restorcon** command to recursively check the files and directories. The **-v** option instructs the command to show any changes made.

Fix the security contexts of the files and directories.

```
# fixfiles -R ruby193-rubygem-passenger restore
# fixfiles -R ruby193-mod_passenger restore
# restorecon -rv /var/run
# restorecon -rv /opt
```

The FQDN for the OpenShift Enterprise domain must be configured. The **CLOUD_DOMAIN** value in the **/etc/openshift/broker.conf** file defines this. Configure the variable to reflect the **example.org** domain.

```
# sed -i -e "s/^CLOUD_DOMAIN=.*/CLOUD_DOMAIN=example.org/"
/etc/openshift/broker.conf
```

The broker application must know how to access the Mongo database. The **/etc/openshift/broker.conf** file defines the **MONGO_DB**, **MONGO_USER**, and the **MONGO_PASSWORD** variables. Set **MONGO_DB** to **openshift_broker**. Set **MONGO_USER** to **openshift**, and set **MONGO_PASSWORD** to the password of the Mongo database OpenShift user.

```
# sed -i '/MONGO_USER/cMONGO_USER="openshift"' /etc/openshift/broker.conf
# sed -i '/MONGO_PASSWORD/cMONGO_PASSWORD="[Password]"'
/etc/openshift/broker.conf
# sed -i '/MONGO_DB/cMONGO_DB="openshift_broker"' /etc/openshift/broker.conf
# grep -i MONGO /etc/openshift/broker.conf

# Eg: MONGO_HOST_PORT="<host1:port1>,<host2:port2>..."
MONGO_HOST_PORT="localhost:27017"
MONGO_USER="openshift"
MONGO_PASSWORD="[Password]"
MONGO_DB="openshift_broker"
MONGO_SSL="false"
```



8.1.7.1 Broker Plugins

OpenShift Enterprise uses plugins for authentication, Dynamic DNS Updates, and messaging. These plugins are disabled by default. Enabling the plugins is done by creating a configuration file for the plugin in the `/etc/openshift/plugins.d` directory. The file name takes the form of `pluginname.conf` and contains sets of key/value pairs.

The three plugins needed are `OPENSIFT-ORIGIN-AUTH-REMOTE-USER`, `OPENSIFT-ORIGIN-MSG-BROKER-MCOLLECTIVE`, and `OPENSIFT-ORIGIN-DNS-NSUPDATE`. Example configuration files exist in the `/etc/openshift/plugins.d` directory. The `OPENSIFT-ORIGIN-AUTH-REMOTE-USER` and `OPENSIFT-ORIGIN-MSG-BROKER-MCOLLECTIVE` plugin configuration files do not need any changes and can be copied to the correct name to enable the function.

Change to the plugin directory and copy the two example plugin files to the correct name.

```
# cd /etc/openshift/plugins.d/
```

Run the following command to copy the example configuration file for the remote user authentication plug-in:

```
# cp openshift-origin-auth-remote-user.conf.example openshift-origin-auth-remote-user.conf
```

Run the following command to copy the example configuration file for the MCollective messaging plug-in:

```
# cp openshift-origin-msg-broker-mcollective.conf.example openshift-origin-msg-broker-mcollective.conf
```

An example configuration file for the `OPENSIFT-ORIGIN-DNS-NSUPDATE` plugin exists. It can be used for reference since it contains comments on the key/value pairs, but a new file is used in this environment.

Create the `OPENSIFT-ORIGIN-DNS-NSUPDATE.CONF` file with the following contents:

```
BIND_SERVER="10.16.136.36"
BIND_PORT="53"
BIND_KEYNAME="example.org"
BIND_KEYVALUE="GpqmTSUeLes1/x8FPQnftdTd4mmzoRPjtLOA6s5ciK0="
BIND_ZONE="example.org"
```

The value for **BIND_KEYVALUE** is the key value from the name servers DNSSEC key.



8.1.7.2 httpd Authentication

The remote-user plugin uses the **httpd** service for authentication. The **httpd** service can be configured to use any of its supported authentication mechanisms, **htpasswd**, **kerberos**, and **ldap**. Sample configuration files are contained in the `/var/www/openshift/broker/httpd/conf.d` directory.

This environment uses a simple *htpasswd* file for authentication. To enable **htpasswd** authentication for the broker, copy the *openshift-origin-auth-remote-user-basic.conf.sample* file to *openshift-origin-auth-remote-user.conf*.

```
# cp /var/www/openshift/broker/httpd/conf.d/openshift-origin-auth-remote-user-basic.conf.sample /var/www/openshift/broker/httpd/conf.d/openshift-origin-auth-remote-user.conf
```

The configuration file specifies `/etc/openshift/htpasswd` for the user authentication file. Use the **htpasswd** command to create the authentication file and add a new user called **rosesuser**. The **-c** option instructs the command to create a new file.

```
# htpasswd -c /etc/openshift/htpasswd rosesuser
```

```
New password: [Password]
```

```
Re-type new password: [Password]
```

```
Adding password for user rosesuser
```

8.1.7.3 Bundler

Bundler is used to track ruby modules and versions. This helps ensure the Broker application functions correctly. Execute the bundler using the **scl** command. The **scl** command is used to run software that is packaged in a software collection.

```
# cd /var/www/openshift/broker
```

```
# scl enable ruby193 'bundle --local'
```

```
Using rake (0.9.2.2)
```

```
Using bigdecimal (1.1.0)
```

```
Using i18n (0.6.0)
```

```
[ ... Output Abbreviated ... ]
```

```
Using stomp (1.1.8)
```

```
Using systemu (2.5.2)
```

```
Using openshift-origin-msg-broker-mcollective (1.9.9)
```

```
Using parseconfig (1.0.2)
```

```
Using regin (0.3.7)
```

```
Using xml-simple (1.0.12)
```

```
Your bundle is complete! Use `bundle show [gemname]` to see where a bundled gem is installed.
```



Enable the broker application service, **openshift-broker**, to start on boot and start the service.

```
# chkconfig openshift-broker on

# service openshift-broker start
Starting openshift-broker:
```

[OK]

Use the **curl** command to query the brokers REST API. A status of *200* should be returned if the broker is functioning correctly.

```
# curl -Ik https://localhost/broker/rest/api
```

```
HTTP/1.1 200
Date: Fri, 06 Sep 2013 21:00:45 GMT
Server: Apache/2.2.15 (Red Hat)
X-Powered-By: Phusion Passenger (mod_rails/mod_rack) 3.0.21
X-UA-Compatible: IE=Edge,chrome=1
ETag: "d8d9a2d967545a75401d415c849a571d"
Cache-Control: max-age=0, private, must-revalidate
X-Request-Id: facd8b09bd5977dd3898f58c8bd40894
X-Runtime: 0.008173
X-Rack-Cache: miss
Status: 200
Content-Type: application/json; charset=utf-8
Connection: close
```

The **oo-accept-broker** command checks if the broker host is in a known good state. The **-v** option displays the items being checked. Execute the **oo-accept-broker** command.

```
# oo-accept-broker
```

```
NOTICE: SELinux is Enforcing
NOTICE: SELinux is Enforcing
PASS
```



8.2 Node Host

Create a virtual machine to act as a node host. Use a name of roses-node and the information in **Table 3.2: OSE Virtual Machine Configuration** to create the virtual machine.

Perform a basic installation of Red Hat Enterprise Linux 6.4.

8.2.1 Network Configuration

Configure the host networking as described in **Table 8.2.1.1: roses-node Network Configuration**.

Setting	Value
Hostname	roses-node.example.org
IP Address	10.16.136.74
Network Mask	255.255.248.0
Gateway	10.16.143.254
Name Resolution	10.16.136.36

Table 8.2.1.1: roses-node Network Configuration



8.2.2 Updates

Register the virtual machine to receive updates and subscribe the host to the appropriate channels to install the node software. Use **Table 8.2.2.1: Node Host Entitlements** as a reference.

Channel	Subscription Manager	RHN Classic
Red Hat OpenShift Enterprise 1.2 Application Node	OpenShift Enterprise	rhel-x86_64-server-6-ose-1.2-node
Red Hat OpenShift Enterprise 1.2 JBoss EAP add-on	JBoss Enterprise Application Platform for OpenShift Enterprise	rhel-x86_64-server-6-ose-1.2-jbosseap
JBoss Enterprise Application Platform	JBoss Enterprise Application Platform for OpenShift Enterprise	jbappplatform-6-x86_64-server-6-rpm
JBoss Enterprise Web Server 2	OpenShift Enterprise	jb-ews-2-x86_64-server-6-rpm
Red Hat Enterprise Virtualization Guest Agent	Red Hat Enterprise Linux Server	rhel-x86_64-rhev-agent-6-server
Software Collection		rhel-x86_64-server-6-rhsc1-1

Table 8.2.2.1: Node Host Entitlements

Install the Red Hat Enterprise Virtualization guest agent for the virtual machine.

```
# yum -y install rhvm-guest-agent
# service ovirt-guest-agent start
# chkconfig ovirt-guest-agent on
```

Packages from third-party repositories or other Red Hat products can conflict with the packages installed for OpenShift Enterprise on the node host. Adjusting the priorities of the repositories and excluding certain package updates can prevent this. Download the scripts from <https://github.com/openshift/openshift-extras/tree/enterprise-2.0/admin> execute the `oo-admin-yum-validator` script using the `--role node --role node-eap` options to check for any issues and then to repair the issues. See **Section 8.1.2 Updates** for an explanation of these scripts and how to use them.

```
# ./oo-admin-check-sources.py --report-all --role broker --role client
--repo-config repos.ini
# ./oo-admin-check-sources.py --fix-all --role broker --role client --repo-
config repos.ini
```



Update the packages on the host.

```
# yum -y update
```

8.2.3 Time Synchronization

Enable and start the **ntpd** service to ensure the system has the correct time. Check the date and time of the system and make sure it is correct. Fix the date and time if necessary.

```
# chkconfig ntpd on
```

```
# service ntpd start
```

```
Starting ntpd: [ OK ]
```

8.2.4 MCollective

The **mcollective** service is used to communicate to the broker. The **mcollective** service is installed when the **openshift-origin-msg-node-mcollective** package is installed. Install the package.

```
# yum -y install openshift-origin-msg-node-mcollective
```

The configuration file for the **mcollective** server is `/etc/mcollective/server.cfg`. This file is similar to the `/etc/mcollective/client.cfg` file on the broker host. The `server.cfg` file defines the **daemonize** variable, whereas the `client.cfg` file on the broker does not.

Create the `/etc/mcollective/server.cfg` file using the following contents.

```
topicprefix = /topic/
main_collective = mcollective
collectives = mcollective
libdir = /opt/rh/ruby193/root/usr/libexec/mcollective
logfile = /var/log/mcollective.log
loglevel = debug

daemonize = 1
direct_addressing = 1

# Plugins
securityprovider = psk
plugin.psk = [Pre-Shared Key]

connector = activemq
plugin.activemq.pool.size = 1
plugin.activemq.pool.1.host = roses-broker.example.org
plugin.activemq.pool.1.port = 61613
plugin.activemq.pool.1.user = mcollective
plugin.activemq.pool.1.password = [Password]

# Facts
factsource = yaml
plugin.yaml = /etc/mcollective/facts.yaml
```



Enable the **mcollective** service to start upon boot and start the service.

```
# chkconfig mcollective on

# service mcollective start
Starting mcollective: [ OK ]
```

8.2.5 Node Packages

Install the packages required for the node host.

```
# yum -y install rubygem-openshift-origin-node ruby193-rubygem-passenger-native openshift-origin-port-proxy openshift-origin-node-util
```

8.2.6 Cartridges

Cartridges provide support for various applications such as **ruby**, **perl**, **php**, **postgresql**, and others. The appropriate cartridge must be installed on all node hosts before an application can be developed that uses the cartridge. Use **yum** to list the names of the available cartridges.

```
# yum list "openshift-origin-cartridge*" | grep -o "openshift-origin-cartridge.*noarch"

openshift-origin-cartridge-cron.noarch
openshift-origin-cartridge-diy.noarch
openshift-origin-cartridge-haproxy.noarch
openshift-origin-cartridge-jbosseap.noarch
openshift-origin-cartridge-jbossews.noarch
openshift-origin-cartridge-jenkins.noarch
openshift-origin-cartridge-mock.noarch
openshift-origin-cartridge-mysql.noarch
openshift-origin-cartridge-perl.noarch
openshift-origin-cartridge-php.noarch
openshift-origin-cartridge-postgresql.noarch
openshift-origin-cartridge-python.noarch
openshift-origin-cartridge-ruby.noarch
openshift-origin-cartridge-cron-1.4.noarch
openshift-origin-cartridge-jenkins-client.noarch
openshift-origin-cartridge-mock-plugin.noarch
```

Two *cron* cartridges exist and should not be installed together. Unless specifically required, the **openshift-origin-cartridge-cron-1.4.noarch** cartridge should not be installed. See the [Red Hat knowledge article #479923](#) for more information on this.

Multiple versions of the cartridges may be displayed if the node host is receiving updates from Red Hat Subscription Manager. Care should be taken to ensure the wanted cartridges are installed.



Install all the cartridges except the **openshift-origin-cartridge-cron-1.4.noarch** cartridge.

```
# yum -y install $( yum list "openshift-origin-cartridge*" | grep -o  
"openshift-origin-cartridge.\w*" )
```

8.2.7 SSH Configuration

The broker host connects to the node host using **ssh** to move gears between node hosts if needed. The root public ssh key from the broker host must be placed in the root users *authorized_keys* file on the node host. Add the public key to the *authorized_hosts* file.

```
# mkdir /root/.ssh  
  
# chown -v root.root /root/.ssh  
  
# chmod -v 700 /root/.ssh  
  
# ssh root@roses-broker cat /root/.ssh/rsync_id_rsa.pub >>  
/root/.ssh/authorized_keys  
  
# chmod -v 600 /root/.ssh/authorized_keys
```

Increase the number of allowed concurrent ssh sessions to the node host by appending the following to the */etc/ssh/sshd_config* file.

```
AcceptEnv GIT_SSH  
MaxSessions 40  
MaxStartups 40
```

Restart the **sshd** service.

```
# service sshd restart
```



8.2.8 Service Configuration

Several services are used by the node host. The firewall must allow traffic to pass to these services and the services must be enabled.

```
# lokkit --service=https
# lokkit --service=http
# lokkit --port=8443:tcp
# chkconfig httpd on
# chkconfig oddjobd on
# chkconfig openshift-node-web-proxy on
# service httpd start
# service oddjobd start
# service openshift-node-web-proxy start
```

8.2.9 PAM Configuration

OpenShift Enterprise contains its own **PAM** modules for users that connect to the Node host using **ssh**. The first module, `PAM_OPENSHIFT`, configures the correct SELinux security contexts for the user. The second module, `PAM_NAMESPACE`, configures a private namespace with polyinstantiated directories. Polyinstantiated directories provide a unique `/tmp` and `/dev/shm` directory for each user on the system.

Change the `/etc/pam.d/ssh` PAM module to use the `PAM_OPENSHIFT` module instead of the `PAM_SELINUX` module.

```
# sed -i -e 's|pam_selinux|pam_openshift|g' /etc/pam.d/ssh
```

Modify the `runuser`, `runuser-l`, `sshd`, `su`, and `system-auth-ac` files in the `/etc/pam.d` directory to use the `PAM_NAMESPACE` module by appending the following two lines to each file if they do not already exist.

```
session [default=1 success=ignore] pam_succeed_if.so quiet shell =
/usr/bin/oo-trap-user
session required pam_namespace.so no_unmount_on_close
```



Settings for the polyinstantiated directories are stored in the `/etc/security/namespace.d/tmp.conf` and `/etc/security/namespace.d/shm.conf` files. Create the files with the following contents.

`/etc/security/namespace.d/tmp.conf:`

```
/tmp      $HOME/.tmp/      user:iscript=/usr/sbin/oo-namespace-init root,adm
```

`/etc/security/namespace.d/shm.conf:`

```
/dev/shm tmpfs tmpfs:mntopts=size=5M:iscript=/usr/sbin/oo-namespace-init  
root,adm
```

8.2.10 Cgroup Configuration

OpenShift Enterprise uses cgroups to control the resources available to the application processes.

Modify the `runuser`, `runuser-l`, `sshd`, `su`, and `system-auth-ac` files to use the `PAM_CGROUP` module by appending the following line to each file if it does not already exist.

```
session    optional    pam_cgroup.so
```

The configuration for cgroups is stored in the `/etc/cgconfig.conf` file. This file defines the control groups, parameters and mount points. Edit the file and verify it contains the following contents.

```
mount {  
    cpuset      = /cgroup/cpuset;  
    cpu         = /cgroup/cpu;  
    cpuacct     = /cgroup/cpuacct;  
    memory      = /cgroup/memory;  
    devices     = /cgroup/devices;  
    freezer     = /cgroup/freezer;  
    net_cls     = /cgroup/net_cls;  
    blkio       = /cgroup/blkio;  
}
```

Make sure the file has the correct SELinux security context.

```
# restorecon -rv /etc/cgconfig.conf
```

The control groups are mounted under the `/cgroup` directory. Create this directory and make sure its SELinux security context is correct.

```
# mkdir -p /cgroup  
  
# restorecon -rv /cgroup
```



The control group services must be started in the following order:

1. **cgconfig**
2. **cgred**
3. **openshift-cgroups**

Make sure the services start on boot and start the services.

```
# chkconfig cgconfig on

# chkconfig cgred on

# chkconfig openshift-cgroups on

# service cgconfig restart
Stopping cgconfig service: [ OK ]
Starting cgconfig service: [ OK ]

# service cgred restart
Stopping CGroup Rules Engine Daemon... [ OK ]
Starting CGroup Rules Engine Daemon: [ OK ]

# service openshift-cgroups start [ OK ]
```

8.2.11 Disk Quotas

User disk quotas are used to enforce the storage quotas for each of the defined gears. This is possible since each gear is a user account on the node. The quota limits are stored in the `/etc/openshift/resource_limits.conf` file. The number of files a gear is allowed to own is set using the **quota_files** variable. The amount of storage space a gear is allowed to use is set using the **quota_blocks** variable. Each block equals 1024 bytes.

The application and data for gears is stored in the `/var/lib/openshift` directory. The file system that contains this directory must be mounted with the `USRQUOTA` flag. Use the **df** command to determine which file system contains the `/var/lib/openshift` directory.

```
# df /var/lib/openshift
Filesystem      1K-blocks      Used Available Use% Mounted on
/dev/mapper/myvg-rootvol
                11159472    3315036    7277552   32% /
```

The output of **df** indicates the `/var/lib/openshift` directory is on the `/` (root) file system. Edit the `/etc/fstab` file and add the `USRQUOTA` flag to the mount options for the `/` (root) file system.

```
/dev/mapper/myvg-rootvol / ext4 usrquota,defaults 1 1
```

Remount the `/` (root) file system so the new mount option takes effect.

```
# mount -o remount /
```



User quotas are stored in the *aquota.user* file in the top level directory of the file system in which they are enforced. This file must be created. Use the **quotacheck** command to create the *aquota.user* file for the / (root) file system. The following options are used when running **quotacheck** for the first time.

--create-files	Ignore any existing quota files. Scan the file system and create new quota files.
--no-remount	Do not remount the file system as read-only while scanning it.
--user	Only check user quotas on the specified file systems or in the <i>/etc/mtab</i> file.
--group	Only check group quotas on the specified file systems or in the <i>/etc/mtab</i> file.

Create a new user quota file, make sure it has the correct SELinux security contexts.

```
# quotacheck --create-files --no-remount --user --group /
# restorecon /aquota.user
```

Use the **quotaon** command to enable quotas on the / (root) file system.

```
# quotaon /
```

Check quotas are working using the **repquota** command. Specify the **--all** option to display quotas for all file systems.

```
# repquota --all
*** Report for user quotas on device /dev/mapper/myvg-rootvol
Block grace time: 7days; Inode grace time: 7days
```

		Block limits				File limits			
User		used	soft	hard	grace	used	soft	hard	grace
root	--	3144420	0	0		125495	0	0	
daemon	--	8	0	0		3	0	0	
lp	--	8	0	0		2	0	0	
nobody	--	232	0	0		19	0	0	
abrt	--	24	0	0		4	0	0	
haldaemon	--	8	0	0		2	0	0	
ntp	--	12	0	0		3	0	0	
postfix	--	60	0	0		38	0	0	
rpc	--	12	0	0		4	0	0	
rpcuser	--	16	0	0		5	0	0	
apache	--	12	0	0		3	0	0	
haproxy	--	4	0	0		1	0	0	
postgres	--	16	0	0		4	0	0	
mysql	--	8	0	0		3	0	0	
jboss	--	8732	0	0		1612	0	0	
jenkins	--	12	0	0		3	0	0	
tomcat	--	792	0	0		48	0	0	



8.2.12 SELinux

The SELinux Boolean variables listed in **Table 8.2.12.1: SELinux Booleans for Node host** must be enabled for the node host to function properly.

Boolean	Description
httpd_unified	Allows httpd processes and scripts to read, write, and execute appropriately labeled content.
httpd_can_network_connect	Allows httpd scripts and modules can connect to the network using any port.
httpd_can_network_relay	Allows httpd to act as a relay.
httpd_read_user_content	Allows httpd to read user home directories.
httpd_enable_homedirs	Allow httpd to use user directories in the URL.
httpd_run_stickshift	Allows httpd to run in stickshift mode and not transition to passenger.
allow_polyinstantiation	Enable the use of polyinstantiated directories.

Table 8.2.12.1: SELinux Booleans for Node host

Enable the needed SELinux boolean variables.

```
# setsebool -P httpd_unified=on httpd_can_network_connect=on  
httpd_can_network_relay=on httpd_read_user_content=on  
httpd_enable_homedirs=on httpd_run_stickshift=on allow_polyinstantiation=on
```

After the SELinux booleans are enabled, directories and files must have their SELinux security contexts set correctly. Use the **restorecon** command to set the SELinux security contexts on the `/var/run` and `/var/lib/openshift` directories and the `/usr/sbin/mcollectived`, `/var/log/mcollective.log`, `/var/run/mcollective.pid`, `/etc/openshift/node.conf`, and `/etc/httpd/conf.d/openshift` files.

```
# restorecon -rv /var/run  
  
# restorecon -rv /usr/sbin/mcollectived /var/log/mcollective.log  
/var/run/mcollectived.pid  
  
# restorecon -rv /var/lib/openshift /etc/openshift/node.conf  
/etc/httpd/conf.d/openshift
```



8.2.13 Tuning the Kernel

The tunable kernel parameters listed in **Table 8.2.13.1: Node Tunable Kernel Parameters** must be increased for the node host to function correctly.

Sysctl Setting	Description
kernel.sem	Used to increase the kernel semaphores, allowing more httpd processes
net.ipv4.ip_local_port_range	Used to increase the number of ports available for application proxies
net.netfilter.nf_conntrack_max	Used to increase the size of the connection tracking table

Table 8.2.13.1: Node Tunable Kernel Parameters

Append the following lines to the `/etc/sysctl.conf` to increase the values of the kernel tunables. The values of these kernel parameters may need increased depending on the hardware configuration and system load. These values are a good starting point for initial loads.

```
kernel.sem = 250 32000 32 4096
net.ipv4.ip_local_port_range = 15000 35530
net.netfilter.nf_conntrack_max = 1048576
```

Execute the **sysctl** command so it reloads the new kernel tunables.

```
# sysctl -p /etc/sysctl.conf

net.ipv4.ip_forward = 0
net.ipv4.conf.default.rp_filter = 1
net.ipv4.conf.default.accept_source_route = 0
kernel.sysrq = 0
kernel.core_uses_pid = 1
net.ipv4.tcp_syncookies = 1
kernel.msgmnb = 65536
kernel.msgmax = 65536
kernel.shmmax = 68719476736
kernel.shmall = 4294967296
kernel.sem = 250 32000 32 4096
net.ipv4.ip_local_port_range = 15000 35530
net.netfilter.nf_conntrack_max = 1048576
```



8.2.14 Port Proxy Configuration

The applications contained within gears listen for connections on the loopback interface of the node host only. To accept outside connections, the node host runs a proxy that forwards the request through the loopback interface. For the proxy to function, the appropriate ports must be allowed to pass through the firewall and the proxy and gears services must be started.

Open the firewall ports, enable the proxy and start the proxy and gears.

```
# lokkit --port=35531-65535:tcp

# chkconfig openshift-port-proxy on

# service openshift-port-proxy start

Starting openshift-port-proxy: [ OK ]

# chkconfig openshift-gears on
```

8.2.15 Node Settings

The nodes configuration settings are stored in the `/etc/openshift/node.conf` file. The settings in this file must be changed to match the environment. Edit the `/etc/openshift/node.conf` file and the following variables.

```
PUBLIC_HOSTNAME="roses-node.example.org"
PUBLIC_IP="10.16.136.74"
BROKER_HOST="roses-broker"
CLOUD_DOMAIN="example.org"
```

The `/etc/openshift/env/OPENSHIFT_BROKER_HOST` file must contain the name of the broker host.

```
# echo roses-broker > /etc/openshift/env/OPENSHIFT_BROKER_HOST
```

The `/etc/openshift/env/OPENSHIFT_CLOUD_DOMAIN` file must contain the name of the OpenShift domain.

```
# echo example.org > /etc/openshift/env/OPENSHIFT_CLOUD_DOMAIN
```

The `/etc/httpd/conf.d/000001_openshift_origin_node.conf` file must contain the hostname of the node.

```
# sed -i -e "s/ServerName .*/ServerName `hostname`/" \
> /etc/httpd/conf.d/000001_openshift_origin_node.conf
```



8.2.16 Facter Database

The Facter database stores metadata about the node host. A **cron** job runs every minute to update the facter database. Manually execute the **cron** job to create an initial copy of the Facter database.

```
# /etc/cron.minutely/openshift-facts
```

8.2.17 Reboot and Test

After the node host is configured, reboot to ensure the services start correctly.

```
# init 6
```

After the node host finishes rebooting, it should be checked for functionality. The **oo-accept-node** command checks if the node host is in a known good state. The **-v** option can be used to display the items being checked. Execute the **oo-accept-node** command.

```
# oo-accept-node
```

```
PASS
```

The **mco ping** command is run from the broker node to make sure communications between the nodes and brokers are functioning properly.

```
# mco ping
```

```
roses-node.example.org           time=112.97 ms
```

```
---- ping statistics ----
```

```
1 replies max: 112.97 min: 112.97 avg: 112.97
```

If the **mco ping** command fails, check that the information in the */etc/mcollective/client.conf* file on the broker host and the */etc/mcollective/server.conf* file on the node host contain the correct settings.

Also check that the date and time are correct on both servers. An incorrect date or time prevents the **mco ping** command from working. An entry in the */var/log/mcollective.log* file on the node host indicates a very old message being received is an incorrect date or time is causing issues.

```
W, [2013-11-12T04:17:10.903009 #4544] WARN -- : runner.rb:65:in `rescue in  
block in run' Message 39aca36c6e565f8dafa5ef719c10d0b8 from uid=0@roses-  
broker.example.org created at 1384229823 is 21607 seconds old, TTL is 60
```



9 Creating Gears

Gears can be created and managed using the REST API, management console, or using the command line client. This reference architecture uses the command line client.

The command line client is called **rhc** and is provided by the package called **rhc**. This package is available in the OpenShift Enterprise subscription if using Subscription Manager for updates or in the `rhel-x86_64-server-6-ose-1.2-rhc` channel if using RHC Classic.

Make sure the broker host is subscribed to the `rhel-x86_64-server-6-ose-1.2-rhc` channel and install the **rhc** package if needed.

```
# rhn-channel -a -c rhel-x86_64-server-6-ose-1.2-rhc
Username: admin
Password:

# rhn-channel -l

rhel-x86_64-rhev-agent-6-server
rhel-x86_64-server-6
rhel-x86_64-server-6-ose-1.2-infrastructure
rhel-x86_64-server-6-ose-1.2-rhc

# yum -y install rhc
```

The **setup** directive should be passed to the **rhc** command the first time the command is executed. This configures the environment for use on the local server. This creates an **ssh** key pair and added the public key to the account on the broker as well as creating an initial namespace if needed and clones any GIT repositories.

By default the **rhc** command connects to the servers at the Red Hat OpenShift Online service. The **--server** option allows the specification of a different broker. Execute the **rhc** command for the first time.

```
# rhc setup --server roses-broker.example.org
OpenShift Client Tools (RHC) Setup Wizard

This wizard will help you upload your SSH keys, set your application
namespace, and check that other programs like Git are properly installed.

The server's certificate is self-signed, which means that a secure
connection can't be established to 'roses-broker.example.org'.

You may bypass this check, but any data you send to the server could be
intercepted by others.

Connect without checking the certificate? (yes|no): yes
Login to roses-broker.example.org: rosesuser
Password: *****

OpenShift can create and store a token on disk which allows to you to access
the server without using your password. The key is stored in your home
```



directory and should be kept secret. You can delete the key at any time by running 'rhc logout'.

Generate a token now? (yes|no) **yes**

Generating an authorization token for this client ... lasts about 1 day

Saving configuration to /root/.openshift/express.conf ... done

No SSH keys were found. We will generate a pair of keys for you.

Created: /root/.ssh/id_rsa.pub

Your public SSH key must be uploaded to the OpenShift server to access code.
Upload now? (yes|no) **yes**

Since you do not have any keys associated with your OpenShift account, your new key will be uploaded as the 'default' key.

Uploading key 'default' ... done

Checking for git ... found git version 1.7.1

Checking common problems .. done

Checking your namespace ... none

Your namespace is unique to your account and is the suffix of the public URLs we assign to your applications. You may configure your namespace here or leave it blank and use 'rhc create-domain' to create a namespace later. You will not be able to create applications without first creating a namespace.

Please enter a namespace (letters and numbers only) |<none>|: **planet**
Your domain name 'planet' has been successfully created

Checking for applications ... none

Run 'rhc create-app' to create your first application.

Do-It-Yourself 0.1	rhc create-app <app name> diy-0.1
JBoss Enterprise App Platform 60	rhc create-app <app name> jbosseap-6.0
Jenkins Server 1.4	rhc create-app <app name> jenkins-1.4
Mock Cartridge 0.1	rhc create-app <app name> mock-0.1
Mock Cartridge 0.2	rhc create-app <app name> mock-0.2
PHP 5.3	rhc create-app <app name> php-5.3
Perl 5.10	rhc create-app <app name> perl-5.10
Python 2.6	rhc create-app <app name> python-2.6
Ruby 1.8	rhc create-app <app name> ruby-1.8
Ruby 1.9	rhc create-app <app name> ruby-1.9
Tomcat 6 (JBoss EWS 1.0)	rhc create-app <app name> jbossews-1.0
Tomcat 7 (JBoss EWS 2.0)	rhc create-app <app name> jbossews-2.0

You are using 0 of 100 total gears

The following gear sizes are available to you: small

Your client tools are now configured.



Creating an application is done using the **rhc create-app** command. The word **help** added to the command line displays options for each command if needed.

The **rhc create-app** command requires the name of the application and the cartridge to use when building the application. The **--gear-size** option specifies the size of the gears available. This process clones a **Git** repository from the OpenShift Enterprise host node to the local directory the command is executed from.

Create a PHP application called earth. Use the default gear size.

```
# cd $HOME

# rhc create-app earth php-5.3

Application Options
-----
Namespace: planet
Cartridges: php-5.3
Gear Size: default
Scaling: no

Creating application 'earth' ... done

Waiting for your DNS name to be available ... done

Downloading the application Git repository ...
Initialized empty Git repository in /root/earth/.git/
Warning: Permanently added 'earth-planet.example.org,10.16.136.74' (RSA) to
the list of known hosts.

Your application code is now in 'earth'

earth @ http://earth-planet.example.org/ (uuid: 52447fb615b3c85dbb000006)
-----
Created: 1:40 PM
Gears: 1 (defaults to small)
Git URL: ssh://52447fb615b3c85dbb000006@earth-
planet.example.org/~/.git/earth.git/
SSH: 52447fb615b3c85dbb000006@earth-planet.example.org

php-5.3 (PHP 5.3)
-----
Gears: 1 small

RESULT:
Application earth was created.
```

Open a browser to <http://earth-planet.example.org> and view the web page.



The directory for the cloned GIT repository is called *earth*. Changes to the website are made through this directory and then **git** command is used to push them to the correct node host.

Change to the *earth* directory and list its sub-directories.

```
# cd earth

# ls

deplist.txt  libs/  misc/  php/  README.md
```

The *php* directory contains the scripts for the application. Inside this directory is a file called *index.php*. Replace the contents of this file with the following.

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">
  <title>Welcome to Planet Earth</title>
  <style>
    html {
      background: blue;
    }
    body {
      background: #333;
      color: white;
      font-family: "Helvetica Neue",Helvetica,"Liberation Sans",Arial,sans-
serif;
      width: 40em;
      margin: 0 auto;
      padding: 3em;
    }
  </style>
</head>
<body>
  <h1>
    Welcome to Planet Earth
  </h1>
</body>
</html>
```




Use **git** to commit the changes to the file and then push the changes to the host node.

```
# git commit --all --message "Initial Page Push"
[master 5492b10] Initial Page Push
Committer: root <root@roses-broker.cloud.lab.eng.bos.redhat.com>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly:
```

```
git config --global user.name "Your Name"
git config --global user.email you@example.com
```

If the identity used for this commit is wrong, you can fix it with:

```
git commit --amend --author='Your Name <you@example.com>'
```

```
1 files changed, 26 insertions(+), 156 deletions(-)
rewrite php/index.php (92%)
```

```
# git push
Warning: Permanently added 'earth-planet.example.org,10.16.136.74' (RSA) to
the list of known hosts.
Counting objects: 7, done.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 454 bytes, done.
Total 4 (delta 2), reused 0 (delta 0)
remote: CLIENT_MESSAGE: Stopping Apache+mod_php HTTPD server
remote: Waiting for stop to finish
remote: CLIENT_MESSAGE: Starting Apache+mod_php HTTPD server
To ssh://52458ee815b3c8bfe00000006@earth-planet.example.org/~git/earth.git/
6441ad1..5492b10 master -> master
```



The new application is now available at earth-planet.example.org. Use a browser or the **curl** command to test the page.

```
# curl http://earth-planet.example.org

<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">
  <title>Welcome to Planet Earth</title>
  <style>
  html {
  background: blue;
  }
  body {
  background: #333;
  color: white;
  font-family: "Helvetica Neue",Helvetica,"Liberation Sans",Arial,sans-
serif;
  width: 40em;
  margin: 0 auto;
  padding: 3em;
  }
  </style>
</head>
<body>
  <h1>
  Welcome to Planet Earth
  </h1>
</body>
```

Create a second **php** application called mars. Use the same *index.php* file, changing all occurrences of earth with mars. The second application is available at mars.planet.example.org. Use **curl** to test the application works.



10 High Availability of Node Host

The environment as it is currently configured does not allow a quick recovery of the gears and applications running on a node host. The current environment relies upon a current backup of the gears to be restored on another Node host. Any gears created since the backup would be lost.

One alternative is to place the gear and application data on shared storage that can be easily migrated to another node host. Other directories, files, and configuration must also be placed on the shared storage so that it is available to the new node host. The following is a list of files and directories that contain information that must be available to the new node host.

/var/lib/openshift

Contains the data for the gears and applications. Must also have quota information migrated to the shared storage.

/etc/openshift

Contains the node configuration for OpenShift Enterprise.

/etc/mcollective

Contains the node configuration for mcollective.

/etc/security/namespace.d/tmp.conf

/etc/security/namespace.d/shm.conf

/etc/security/limits.d/\${GEAR}.conf

Contains the PAM process limits for the gear.

/etc/cgrouules.conf

Defines which cgroups a process group belongs to.

/etc/passwd

Need the UID and GID of gear users.

/etc/sysconfig/network

Need the hostname.

/var/named/dynamic/example.org.zone

Need to update the DNS record on roses-ns.example.org



10.1 Preparing the host node for failover

Unless otherwise stated, the commands in this section are executed on the first node host.

A virtual disk must be created to hold the gear data and configuration files. Most of the data stored is gear and application data. This is stored in the `/var/lib/openshift` directory on the node host. Execute the `du` command on the directory to determine the minimum amount of space required for the shared storage.

```
# du --human-readable --summarize /var/lib/openshift
11M  /var/lib/openshift/
```

The gears and applications only utilize about 11 MB of storage. A 20 GB shared storage device is sufficient for storing the gear data, application data, and configuration data in the environment used in this reference architecture. A production system most likely requires a much larger device for the shared storage.

Create the shared storage by logging into the Red Hat Enterprise Virtualization administrative portal. Select **Add** under the **Disks** Tab.

An **Add Virtual Disk** screen appears. Specify a **Size** of **20** GB. Specify **roses-node-gears** for the **Alias**. Make sure **Is shareable** is checked. Select **OK** to create the virtual disk.

Select the **Virtual Machines** tab. Select the **roses-node** virtual machine. Select the **Disks** tab at the bottom of the screen. Select **Add**.

An **Add Virtual Disk** screen appears. Select **Attach** Disk. A list of disks appear. Place a check in the box next to the **roses-node-gears** disk. Select **Activate**. Click **OK** to attach the disk to the virtual machine and activate it.

10.1.1 Prepare Shared Storage

When partitioning the storage, requirements of the shared directories must be taken into consideration.

The `/var/lib/openshift` directory contains the gears and application data. This directory contains the bulk of the data and information needed to bring up the gears and applications on another node host. This directory also requires the use of disk quotas. Because of this requirement, it is much easier to have this directory exist on the shared storage as its own partition. This partition is mounted under `/var/lib/openshift`.

The `/etc/openshift` and `/etc/mcollective` directories are small directories and do not have any special considerations that must be made. These directories can exist as sub-directories on the same partition. These sub-directories are `BIND` mounted onto the appropriate directories. See the **man** page for the **mount** command for a further explanation of `BIND` mounts.

The remaining configuration consists of individual files or configuration within the files. This information is gathered and stored on the same partition as the `/etc/openshift` and `/etc/mcollective` shared directories.



Taking the directories and file into consideration, only two partitions are needed. One partition to contain the gear and application data and one partition to contain the remaining configuration files. The configuration directories and files are relatively small and are less likely to grow in size as the gear and application directories. A 5 GB partition should be fine for the configuration directories and files and still allow them room to grow in size. The remain 15 GB of the shared storage is used for the gear and application data since it has the potential to need more space quickly.

10.1.1.1 Partition and format the shared storage

Use the **parted --list** command to list the disks in the system. The `/dev/vdb` disk does not contain a disk label. This is the newly added shared storage disk.

```
# parted --list

Model: Linux device-mapper (linear) (dm)
Disk /dev/mapper/myvg-rootvol: 11.6GB
Sector size (logical/physical): 512B/512B
Partition Table: loop

Number  Start   End     Size    File system  Flags
  1      0.00B   11.6GB  11.6GB  ext4

Model: Virtio Block Device (virtblk)
Disk /dev/vda: 12.9GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos

Number  Start   End     Size    Type        File system  Flags
  1      1049kB  211MB   210MB   primary     ext3          boot
  2      211MB   1259MB  1049MB  primary     linux-swap(v1)
  3      1259MB  12.9GB  11.6GB  primary

Error: /dev/vdb: unrecognised disk label
```

Use the **parted** command to add an *MSDOS PARTITION* table to the device.

```
# parted --script /dev/vdb "mklabel msdos"
```

Create a 5 GB primary partition for the configuration directories and files. Start the partition at 1024 bytes.

```
# parted --script /dev/vdb "mkpart primary 1024 5G"
```

Create a second partition for the gear and application data. Configure the partition to use the remaining space on the device by specifying `-1s` as the last sector for the partition.

```
# parted --script /dev/vdb "mkpart primary 5G -1s"
```



View the newly created partition table.

```
# parted --script /dev/vdb "print"
```

```
Model: Virtio Block Device (virtblk)
Disk /dev/vdb: 21.5GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
```

Number	Start	End	Size	Type	File system	Flags
1	1024MB	5000MB	3975MB	primary		
2	5000MB	21.5GB	16.5GB	primary		

Create an ext4 file system on the partitions.

```
# mkfs.ext4 /dev/vdb1
```

```
# mkfs.ext4 /dev/vdb2
```

Create a directory for use as a mount point in the */var/lib* directory called *node_share*. This mount point is used to mount the file system that contains the configuration directories and files.

```
# mkdir /var/lib/node_share
```

The */var/lib/openshift* directory already exists and does not need to be created.



10.1.2 Move Directories to Shared Storage

Before the data and configuration files can be moved to the shared storage, the services that use them must be stopped.

Stop the services.

```
# for i in mcollective \  
> httpd \  
> ntpd \  
> oddjobd \  
> openshift-node-web-proxy \  
> cgconfig \  
> cgred \  
> openshift-cgroups \  
> openshift-port-proxy \  
> openshift-gears \  
> do \  
> service ${i} stop \  
> done  
  
Shutting down mcollective:           [ OK ]  
Stopping httpd:                       [ OK ]  
Shutting down ntpd:                   [ OK ]  
Shutting down oddjobd:                 [ OK ]  
Stopping node-web-proxy:               [ OK ]  
Stopping cgconfig service:             [ OK ]  
Stopping CGroup Rules Engine Daemon... [ OK ]  
Stopping openshift-port-proxy:         [ OK ]  
Stopping gear 526828bb325710552e00003c... [ OK ]  
Stopping gear 526828da325710552e000051... [ OK ]
```

After the services are stopped, the configuration files and directories can be copied to the first partition of the shared storage device. Mount the first partition of the shared device under `/var/lib/node_share`.

```
# mount /dev/vdb1 /var/lib/node_share
```



Once the partition is mounted, the contents of the `/etc/openshift` and `/etc/mcollective` directories can be moved to the shared storage. Directories must be created to store the configuration files. Create two directories under `/var/lib/node_share` called `etc_openshift` and `etc_mcollective`. The directories contain hidden files that begin with a dot.

By default, the shell does not match these files when an asterisk (*) is used for file name expansion. Using the **shopt** command to enable the **dotglob** option changes this behavior so that expansion matches files beginning with a dot. This allows the use of a single **mv** command to move all the directory contents.

```
# mkdir /var/lib/node_share/etc_openshift
# mkdir /var/lib/node_share/etc_mcollective
# shopt -s dotglob
# mv /etc/openshift/* /var/lib/node_share/etc_openshift
# mv /etc/mcollective/* /var/lib/node_share/etc_mcollective
```

After the configuration files are moved to the new location, the directories can be *BIND* mounted to their original places. In order to do this, the SELinux security contexts of the new directories must be set correctly. The **chcon** command can be used to copy the SELinux security contexts from the directory referenced by the **--reference** option to the new directory.

```
# chcon --reference /etc/openshift /var/lib/node_share/etc_openshift
# chcon --reference /etc/mcollective /var/lib/node_share/etc_mcollective
```

These directories and the partition must be mounted upon boot. Add entries in the `/etc/fstab` to mount the partition and *BIND* mount the directories.

<code>/dev/vdb1</code>	<code>/var/lib/node_share</code>	<code>ext4</code>	<code>defaults</code>
<code>/var/lib/node_share/etc_openshift</code>	<code>/etc/openshift</code>	<code>none</code>	<code>bind,auto</code>
<code>/var/lib/node_share/etc_mcollective</code>	<code>/etc/mcollective</code>	<code>none</code>	<code>bind,auto</code>

Test the *fstab* entries by mounting the directories.

```
# mount /etc/openshift
# mount /etc/mcollective
```

Execute the mount command to verify the directories are mounted with the **bind** option.

```
# mount | grep /etc
/var/lib/node_share/etc_openshift on /etc/openshift type none (rw,bind)
/var/lib/node_share/etc_mcollective on /etc/mcollective type none (rw,bind)
```




The second partition on the shared storage device must be mounted so the gear and application data can be moved to it. Since the mount point is also the location of the data, the partition must be mounted elsewhere. Mount the partition on */mnt*.

```
# mount /dev/vdb2 /mnt
```

Move the gear and application data from */var/lib/openshift* to the second partition of the shared storage device.

```
# mv /var/lib/openshift/* /mnt
```

After the data is moved, set the correct SELinux security context and unmount the partition.

```
# chcon --reference /var/lib/openshift /mnt
```

```
# umount /mnt
```

Create an entry in the */etc/fstab* file to enable the partition to be mounted upon boot. This file system also requires user disk quotas. Specify the *USRQUOTA* option as a mount option.

```
/dev/vdb2 /var/lib/openshift ext4 usrquota,defaults
```

Test the *fstab* entry by mounting the directory.

```
# mount /var/lib/openshift
```

Create a new user quota file, make sure it has the correct SELinux security contexts.

```
# quotacheck --create-files --no-remount --user --group /var/lib/openshift
```

```
# restorecon /var/lib/openshift/aquota.user
```

Use the **quotaon** command to enable quotas on the */var/lib/openshift* file system.

```
# quotaon /var/lib/openshift
```



The quotas for the gear users must be added to the `/var/lib/openshift/aquota.user` file. Each gear user has a directory with the same name under `/var/lib/openshift`. View the directories in `/var/lib/openshift` to determine the gear users. Run the **repquota** command on the `/` (root) file system to determine the quota settings for each gear user.

```
# ls /var/lib/openshift

526828bb325710552e00003c 526828da325710552e000051 aquota.user lost+found

# repquota /

*** Report for user quotas on device /dev/mapper/myvg-rootvol
Block grace time: 7days; Inode grace time: 7days
```

User	used	Block limits		grace	used	File limits		grace
		soft	hard			soft	hard	
root	-- 3125700	0	0		124960	0	0	
daemon	-- 8	0	0		3	0	0	
lp	-- 8	0	0		2	0	0	
nobody	-- 232	0	0		19	0	0	
abrt	-- 20	0	0		4	0	0	
haldaemon	-- 8	0	0		2	0	0	
ntp	-- 12	0	0		3	0	0	
postfix	-- 60	0	0		38	0	0	
rpc	-- 12	0	0		4	0	0	
rpcuser	-- 16	0	0		5	0	0	
ovirtagent	-- 12	0	0		3	0	0	
apache	-- 28	0	0		7	0	0	
haproxy	-- 4	0	0		1	0	0	
postgres	-- 16	0	0		4	0	0	
mysql	-- 8	0	0		3	0	0	
jboss	-- 8732	0	0		1612	0	0	
jenkins	-- 12	0	0		3	0	0	
tomcat	-- 792	0	0		48	0	0	
526828bb325710552e00003c	--	0	0	0	1048576			1 0
40000								
526828da325710552e000051	--	0	0	0	1048576			1 0
40000								



Use the **setquota** command to add the quotas to the */var/lib/openshift* file system. The **setquota** command requires the **--user** option followed by the gear name, quota values, and file system. The quota values are listed in the following order separated by spaces: block-softlimit block-hardlimit inode-softlimit inode-hardlimit.

```
# setquota --user 526828bb325710552e00003c 0 1048576 0 1048576
/var/lib/openshift

# setquota --user 526828da325710552e000051 0 1048576 0 40000
/var/lib/openshift

# repquota /var/lib/openshift

*** Report for user quotas on device /dev/vdb2
Block grace time: 7days; Inode grace time: 7days
      Block limits
User      used      soft      hard      grace      used      soft      hard      grace
-----
root      --      9068      0      0      991      0      0
526828bb325710552e00003c --      780      0 1048576      168      0
1048576
526828da325710552e000051 --      780      0 1048576      168      0
40000
```



After the configuration directories, files, and data are moved to the shared storage, the services can be restarted.

```
# for i in mcollective \
> httpd \
> ntpd \
> oddjobd \
> openshift-node-web-proxy \
> cgconfig \
> cgred \
> openshift-cgroups \
> openshift-port-proxy \
> openshift-gears
> do
> service ${i} start
> done

Starting mcollective: [ OK ]
Starting httpd:
[Wed Oct 23 15:10:14 2013] [warn] module passenger_module is already loaded,
skipping
[ OK ]
Starting ntpd: [ OK ]
Starting oddjobd: [ OK ]
Starting node-web-proxy: [ OK ]
Starting cgconfig service: [ OK ]
Starting CGroup Rules Engine Daemon: [ OK ]
Starting openshift-port-proxy: [ OK ]
Background start initiated - process id = 15473
Check /var/log/openshift/node/platform.log for more details.

Note: In the future, if you wish to start the OpenShift services in the
foreground (waited), use: service openshift-gears waited-start

Starting gear 526828bb325710552e00003c... [ OK ]
Starting gear 526828da325710552e000051... [ OK ]
```

10.1.3 Copy Files and Other Information to Shared Storage

The remaining configuration files and information is copied to the first partition of the shared storage. Copying is performed since only partial directories or configuration files are needed and there is not a clean way to mount this partial information. This information is stored on the shared storage and added to the new node host when needed.



10.1.3.1 Pam Limits

Gear, or user, configuration for the `PAM_LIMITS` module is stored in files in the `/etc/security/limits.d` directory. The files need to be available on the shared storage. Once failover occurs, these files are copied from shared storage to the new host node.

```
# ls /etc/security/limits.d
84-526828bb325710552e00003c.conf  90-nproc.conf
84-526828da325710552e000051.conf
```

Create a directory on the first partition of the shared storage node called `/var/lib/node_share/pam_limits`. This directory is used to store the configuration files for the `PAM_LIMITS` module.

```
# mkdir /var/lib/node_share/pam_limits
```

Copy the `PAM_LIMITS` configuration files for each gear into the `/var/lib/node_share/pam_limits` directory.

```
# for user in $( /bin/ls /var/lib/openshift )
> do
>   if ls -dZ /var/lib/openshift/${user} | grep -q openshift_var_lib_t
>   then
>     cp -v /etc/security/limits.d/*${user}.conf /var/lib/node_share/pam_limits
>   fi
> done
```

10.1.3.2 User IDs and Control Groups

Some information about the users must be gathered. This information is used to create the users on the second node host once it is installed. Only the user id (UID) and group id (GID) are required to create the user accounts. The remaining account information is available from the already captured configuration files.

The control groups each user belongs to must also be captured. This information is stored in the `/etc/cgrouper.conf` file.

This information is gathered and written to the `/var/lib/node_share/node_user_info.current` file in a format that is easily parsed.

```
# for user in $( /bin/ls /var/lib/openshift )
> do
>   if ls -dZ /var/lib/openshift/${user} | grep -q openshift_var_lib_t
>   then
>     t_uinfo=$( grep ${user} /etc/passwd | awk -F ':' '{print $3 " " $4}' )
>     t_cgrops=$( grep ${user} /etc/cgrouper.conf | awk '{print $2}' )
>
>     echo "${user} ${t_uinfo} ${t_cgrops}"
>   fi
> done > /var/lib/node_share/node_user_info.current
```



10.1.3.3 Cron Job

Gears can be added and removed from a node at any time. When this happens, `PAM_LIMIT` configuration files, user information, and cgroup configuration changes. The steps above are placed in a script that can be executed within **cron**.

Create a directory called `/etc/cron.5minutely`. Set the correct permissions and SELinux security contexts.

```
# mkdir /etc/cron.5minutely

# chmod -v 755 /etc/cron.5minutely

# chcon --reference /etc/cron.hourly /etc/cron.5minutely
```

Create the `/etc/cron.5minutely/gear_user_info.sh` file with the following contents.

```
#!/bin/bash

if [ ! -d /var/lib/node_share/pam_limits ]
then
    mkdir /var/lib/node_share/pam_limits
fi

> /var/lib/node_share/node_user_info.current

for user in $( /bin/ls /var/lib/openshift )
do
    if ls -dZ /var/lib/openshift/${user} | grep -q openshift_var_lib_t
    then
        cp -fv /etc/security/limits.d/*${user}.conf /var/lib/node_share/pam_limits

        t_uinfo=$( grep ${user} /etc/passwd | awk -F ':' '{print $3 " " $4}' )
        t_cgrps=$( grep ${user} /etc/cgrules.conf | awk '{print $2}' )

        echo "${user} ${t_uinfo} ${t_cgrps}" >>
/var/lib/node_share/node_user_info.current
        echo "${user} ${t_uinfo} ${t_cgrps}" >>
/var/lib/node_share/node_user_info.$( date +%Y%m )
    fi
done
```

Set the execute bit on the file.

```
# chmod -v +x /etc/cron.5minutely/gear_user_info.sh
```

The example cron script provided above is used to gather the information used in this reference architecture. This script is for example only and has not been tested in a production environment. Ideally this script should make sure accurate user information is written to the files and possible backup the data instead of deleting it. Use of this script in a production environment in its current form is not recommended.



Configure **cron** to run any files in the */etc/cron.5minutely* directory every 5 minutes. Create a file called */etc/cron.d/5minutely*, place the following content in it.

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
HOME=/

*/5 * * * * root run-parts /etc/cron.5minutely
```

Make sure the SELinux security context is correct on the file and tell **crond** to reload its configuration.

```
# chcon --reference /etc/cron.d/sysstat /etc/cron.d/5minutely

# service crond reload

Reloading crond: [ OK ]
```

The example cron job used in this reference architecture is set to run every 5 minutes. This interval may not be sufficient on a production system. Failure to gather the user information can result in lost applications.



11 Failing the Node Host

11.1 Backup Node Host Installation And Configuration

The installation and preparation of the backup node host should be done prior to its need. This speeds up the failover process and increases the up time of the environment.

Create a virtual machine to use as a backup node host. Use the information in **Table 3.2: OSE Virtual Machine Configuration** to create the virtual machine, use *roses-backup-node* for the name.

Install the virtual machine with a basic installation of Red Hat Enterprise Linux 6.4.

11.1.1 Network Configuration

Configure the networking as described in **Table 11.1.1.1: roses-backup-node Network Configuration**.

Setting	Value
Hostname	roses-backup-node.example.org
IP Address	10.16.136.177
Network Mask	255.255.248.0
Gateway	10.16.143.254
Name Resolution	10.16.136.36

Table 11.1.1.1: roses-backup-node Network Configuration

11.1.2 Time Synchronization

Enable and start the **ntpd** service to ensure the system has the correct time.

```
# chkconfig ntpd on
```

11.1.3 Updates

Register the virtual machine and subscribe it to the appropriate channels as listed in **Table 8.2.2.1: Node Host Entitlements**. After the system is registered, install the Red Hat Enterprise Virtualization guest agent for the virtual machine.

```
# yum -y install rhvm-guest-agent
```

```
# chkconfig ovirt-guest-agent on
```




Packages from third-party repositories or other Red Hat products can conflict with the packages installed for OpenShift Enterprise on the node host. Adjusting the priorities of the repositories and excluding certain package updates can prevent this. Download the scripts from <https://github.com/openshift/openshift-extras/tree/enterprise-2.0/admin> execute the `oo-admin-yum-validator` script using the `--role node --role node-eap` options to check for any issues and then to repair the issues. See **Section 8.1.2 Updates** for an explanation of these scripts and how to use them.

```
# ./oo-admin-check-sources.py --report-all --role broker --role client
--repo-config repos.ini

# ./oo-admin-check-sources.py --fix-all --role broker --role client --repo-
config repos.ini
```

Update the packages on the host. Reboot the host to make sure the latest packages are being used.

```
# yum -y update
```

Install all the necessary packages.

```
# yum -y install openshift-origin-msg-node-mcollective rubygem-openshift-
origin-node ruby193-rubygem-passenger-native openshift-origin-port-proxy
openshift-origin-node-util
```

Install the cartridges.

```
# yum -y install $( yum list "openshift-origin-cartridge*" | grep -o
"openshift-origin-cartridge.\w*" )
```

Configure PAM to use the OpenShift Enterprise PAM modules.

```
# sed -i -e 's|pam_selinux|pam_openshift|g' /etc/pam.d/sshd

# for f in "runuser" "runuser-l" "sshd" "su" "system-auth-ac"
> do
>   t="/etc/pam.d/$f"
>   if ! grep -q "pam_namespace.so" "$t"
>   then
>     echo -e "session\t\t[default=1 success=ignore]\tpam_succeed_if.so
quiet shell = /usr/bin/oo-trap-user" >> "$t"
>     echo -e "session\t\t\trequired\tpam_namespace.so no_unmount_on_close" >>
"$t"
>   fi
> done
```



Configure polyinstantiation.

```
# echo "/tmp          \${HOME}/.tmp/          user:iscript=/usr/sbin/oo-namespace-  
init root,adm" > /etc/security/namespace.d/tmp.conf  
  
# echo "/dev/shm tmpfs tmpfs:mntopts=size=5M:iscript=/usr/sbin/oo-  
namespace-init root,adm" > /etc/security/namespace.d/shm.conf
```

Configure cgroups.

```
# for f in "runuser" "runuser-1" "sshd" "system-auth-ac"  
> do  
>   t="/etc/pam.d/$f"  
>   if ! grep -q "pam_cgroup" "$t"  
>   then  
>     echo -e "session\t\ttoptional\tpam_cgroup.so" >> "$t"  
>   fi  
> done  
  
# cp /etc/cgconfig.conf{,orig}  
  
# cp -vf /opt/rh/ruby193/root/usr/share/gems/doc/openshift-origin-node-  
*/cgconfig.conf /etc/cgconfig.conf  
  
# restorecon -rv /etc/cgconfig.conf  
  
# mkdir -p /cgroup  
  
# restorecon -rv /cgroup
```

Configure SELinux.

```
# setsebool -P httpd_unified=on httpd_can_network_connect=on  
httpd_can_network_relay=on httpd_read_user_content=on  
httpd_enable_homedirs=on httpd_run_stickshift=on allow_polyinstantiation=on  
  
# restorecon -rv /var/run  
  
# restorecon -rv /usr/sbin/mcollectived
```



Configure SSH.

```
# mkdir /root/.ssh

# chown -v root.root /root/.ssh

# chmod -v 700 /root/.ssh

# ssh root@roses-broker cat /root/.ssh/rsync_id_rsa.pub >>
/root/.ssh/authorized_keys

# chmod -v 600 /root/.ssh/authorized_keys

# cat << EOSSH >> /etc/ssh/sshd_config
AcceptEnv GIT_SSH
MaxSessions 40
MaxStartups 40
EOSSH

# service sshd restart
```

Tune the sysctl file.

Append the following lines to the sysctl file.

```
kernel.sem = 250 32000 32 4096
net.ipv4.ip_local_port_range = 15000 35530
net.netfilter.nf_conntrack_max = 1048576
```

Tell sysctl to reload the file.

```
# sysctl -p /etc/sysctl.conf
```

The OpenShift Enterprise services do not need to run on the backup node host until it is fully configured. Make sure the OpenShift Enterprise services are stopped.

```
# for svc in openshift-gears openshift-port-proxy openshift-cgroups cgroupd
cgconfig openshift-node-web-proxy oddjobd httpd mcollective
> do
>   service ${svc} stop
> done
```

Configure the */etc/fstab* to mount the file systems on the shared storage upon boot. Add the following lines to the */etc/fstab* file.

```
/dev/vdb1 /var/lib/node_share ext4 defaults
/var/lib/node_share/etc/openshift /etc/openshift none bind,auto
/var/lib/node_share/etc/mcollective /etc/mcollective none bind,auto
/dev/vdb2 /var/lib/openshift ext4 usrquota,defaults
```

Create the */var/lib/node_share* directory to mount the */dev/vdb1* device on.

```
# mkdir /var/lib/node_share
```



The `/etc/openshift`, `/etc/mcollective`, and `/var/lib/openshift` directories should not contain any configuration and files. This helps prevent confusion if mounting a file system fails.

Clear the contents of the directories.

```
# rm -rf /etc/openshift/* /etc/mcollective/* /var/lib/openshift/*
```

Configure firewall settings.

```
# lokkit --service=https
# lokkit --service=http
# lokkit --port=8443:tcp
# lokkit --port=35531-65535:tcp
```

The backup node host is now configured and ready to be used in the event of a failure.

11.2 Power Off the Failed Node Host

Before moving the shared storage to the new node host, the primary node host must be powered off. This is necessary for the integrity of the data on the shared volume. If two systems try to maintain the file system metadata of an ext4 file system at the same time, corruption of the meta data occurs which can result in lost data.

Power off the `roses-node` virtual machine by clicking the right mouse button on the server under the **Virtual Machines** tab in the Red Hat Enterprise Virtualization Manager administrative portal and selecting **Power Off**.

After the node is powered off, select the **Disks** tab at the bottom of the screen. Select the **roses-node-gears** virtual disk and select **Deactivate**. This deactivates the disk so the virtual machine operating system does not see it if it gets accidentally powered on. The disk could be removed from the virtual machine to prevent accidentally activating it, but this is not necessary.



11.3 Power On the Backup Node Host

The first node host `roses-node` should be powered off. The commands in this section should be performed on the second node host `roses-backup-node`.

11.3.1 Attach Shared Storage

After the shared virtual disk is deactivated or detached from the `roses-node` virtual machine, it can be attached and activated on the **roses-backup-node** virtual machine.

Select the **roses-backup-node** virtual machine under the **Virtual Machines** tab in the Red Hat Enterprise Virtualization Manager administrative portal.

Select the **Disks** tab at the bottom of the screen. Select **Add**.

An **Add Virtual Disk** screen appears. Select **Attach Disk**. A list of virtual disks that are available to be attached to the virtual machine are displayed. Select the **roses-node-gears** virtual disk. Select the **activate** check box. Select **OK**.

The **roses-node-gears** virtual disk appears in the **Disks** tab and is activated.

Power on the **roses-backup-node** virtual machine by clicking the right mouse button on the server under the **Virtual Machines** tab in the Red Hat Enterprise Virtualization Manager administrative portal and selecting **Run**.

11.3.2 Mount File Systems

Mount the file systems.ca

```
# mount /var/lib/node_share
# mount /etc/openshift
# mount /etc/mcollective
# mount /var/lib/openshift
```

Make sure quotas are enabled.

```
# quotaon /var/lib/openshift
```



11.3.3 Update Networking

After the new node host is brought up, the host name and the IP address of the host must be changed to reflect the IP of the first node host. This allows proper access to the applications and gears. Change the network information of the backup node host to reflect the information in **Table 8.2.1.1: roses-node Network Configuration**.

An update to the name server records changing the resolution of roses-node.example.org to the IP address of roses-backup-node.example.org cannot be done. Doing so, could break scaled applications since such applications communicate using IP addresses instead of name resolution.

11.3.4 Create User Accounts

Create the needed user accounts for the gears.

```
# while read id uid gid cg
> do
>   groupadd --gid ${gid} ${id}
>   useradd --base-dir /var/lib/openshift -c "OpenShift guest" --gid ${gid}
--uid ${uid} -M -s /var/lib/openshift ${id}

>   echo "${id}      ${cg}      /openshift/${id}" >> /etc/cgrules.conf

>   cp /var/lib/node_share/pam_limits/*${id}.conf /etc/security/limits.d
> done < /var/lib/node_share/node_user_info.current

# restorecon /etc/security/limits.d/*
```

11.3.5 Cron Configuration

Configure *cron* to continue gathering user information in case we need to fail back to the original node or a different node. Create a directory to hold the cron file.

```
# mkdir /etc/cron.5minutely

# chmod -v 755 /etc/cron.5minutely

# chcon --reference /etc/cron.hourly /etc/cron.5minutely
```



Create the `/etc/cron.5minutely/gear_user_info.sh` file with the following contents.

```
#!/bin/bash

if [ ! -d /var/lib/node_share/pam_limits ]
then
    mkdir /var/lib/node_share/pam_limits
fi

> /var/lib/node_share/node_user_info.current

for user in $( /bin/ls /var/lib/openshift )
do
    if ls -dZ /var/lib/openshift/${user} | grep -q openshift_var_lib_t
    then
        cp -fv /etc/security/limits.d/*${user}.conf /var/lib/node_share/pam_limits

        t_uinfo=$( grep ${user} /etc/passwd | awk -F ':' '{print $3 " " $4}' )
        t_cgrps=$( grep ${user} /etc/cgroues.conf | awk '{print $2}' )

        echo "${user} ${t_uinfo} ${t_cgrps}" >>
/var/lib/node_share/node_user_info.current
        echo "${user} ${t_uinfo} ${t_cgrps}" >>
/var/lib/node_share/node_user_info.$( date +%Y%m )
    fi
done
```

Set the execute bit on the file.

```
# chmod -v +x /etc/cron.5minutely/gear_user_info.sh
```

Create a file called `/etc/cron.d/5minutely` that contains the following contents.

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
HOME=/

*/5 * * * * root run-parts /etc/cron.5minutely
```

Tell **cron** to reload its configuration.

```
# service crond reload
```

```
Reloading crond: [ OK ]
```

11.3.6 Factor Database

Rebuild the factor database.

```
# /etc/cron.minutely/openshift-facts
```



11.3.7 Enable Services and Reboot

Enable the services then reboot the system.

```
# for svc in mcollective httpd oddjobd openshift-node-web-proxy cgconfig
cgred openshift-cgroups openshift-port-proxy openshift-gears
> do
>   chkconfig ${svc} on
> done

# init 6
```

11.4 Testing

Verify the OpenShift Enterprise environment is still working after failover, create a new application called neptune.

```
# cd $HOME

# rhc create neptune php-5.3

Application Options
-----
Namespace:  planet
Cartridges: php-5.3
Gear Size:  default
Scaling:    no

[ ... Output Abbreviated ... ]

Application neptune was created.
```




Replace the contents of the `$HOME/neptune/php/index.php` file with the following.

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">
  <title>Welcome to Planet Neptune</title>
  <style>
    html {
      background: blue;
    }
    body {
      background: #333;
      color: white;
      font-family: "Helvetica Neue",Helvetica,"Liberation Sans",Arial,sans-
serif;
      width: 40em;
      margin: 0 auto;
      padding: 3em;
    }
  </style>
</head>
<body>
  <h1>
    Welcome to Planet Neptune
  </h1>
</body>
</html>
```

Push the new `index.php` file.

```
# cd neptune
# git commit --all --message "Initial Page Push"
# git push
```

Verify the new application is working. The output shows the three applications are running.

```
# curl -s http://earth-planet.example.org http://mars-planet.example.org
http://neptune-planet.example.org | grep title
<title>Welcome to Planet Earth</title>
<title>Welcome to Planet Mars</title>
<title>Welcome to Planet Neptune</title>
```

The applications and gears have successfully been brought up on the backup node host.



12 Failing Red Hat Storage Nodes

When a Red Hat Storage node fails, the virtual machine in the Red Hat Enterprise Virtualization environment are not affected since the storage bricks are replicated between the storage nodes.

12.1 Verify Applications

Make sure the applications are functioning.

```
# curl -s http://earth-planet.example.org http://mars-planet.example.org
http://neptune-planet.example.org | grep title

<title>Welcome to Planet Earth</title>
<title>Welcome to Planet Mars</title>
<title>Welcome to Planet Neptune</title>
```

Perform a hard power off of the ra-rhs-srv3 storage node and verify the applications still function.

```
# curl -s http://earth-planet.example.org http://mars-planet.example.org
http://neptune-planet.example.org | grep title

<title>Welcome to Planet Earth</title>
<title>Welcome to Planet Mars</title>
<title>Welcome to Planet Neptune</title>
```

12.2 Create an Application

Create a new application in the OpenShift Enterprise environment called jupiter. Use the roses-broker host to do this.

```
# cd $HOME

# rhc create-app jupiter php-5.3

Your authorization token has expired. Please sign in now to continue.
Password: *****

Application Options
-----
Namespace: planet
Cartridges: php-5.3
Gear Size: default
Scaling: no

[ ... Output Abbreviated ... ]

Application jupiter was created.
```



Replace the contents of the `$HOME/jupiter/php/index.php` file with the following.

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">
  <title>Welcome to Planet Jupiter</title>
  <style>
    html {
      background: blue;
    }
    body {
      background: #333;
      color: white;
      font-family: "Helvetica Neue",Helvetica,"Liberation Sans",Arial,sans-
serif;
      width: 40em;
      margin: 0 auto;
      padding: 3em;
    }
  </style>
</head>
<body>
  <h1>
    Welcome to Planet Jupiter
  </h1>
</body>
</html>
```

Push the new `index.php` file.

```
# cd jupiter
# git commit --all --message "Initial Page Push"
# git push
```

Verify the new application is working.

```
# curl -s http://earth-planet.example.org http://mars-planet.example.org
http://neptune-planet.example.org http://jupiter-planet.example.org | grep
title

<title>Welcome to Planet earth</title>
<title>Welcome to Planet mars</title>
<title>Welcome to Planet Neptune</title>
<title>Welcome to Planet Jupiter</title>
```



12.3 Healing the Storage Files

The files that are out of sync, need healed, between the Red Hat Storage nodes are viewed using the **gluster volume heal ROSESdatadomain info** command. Issue the command on the ra-rhs-srv4 host.

```
# gluster volume heal ROSESdatadomain info

Gathering list of entries to be healed on volume ROSESdatadomain has been
successful

Brick ra-rhs-srv3-
10g.storage.se.priv:/.rhs/ROSESdatadomain/ROSESdatadomain_brick
Number of entries: 0

Brick ra-rhs-srv4-
10g.storage.se.priv:/.rhs/ROSESdatadomain/ROSESdatadomain_brick
Number of entries: 7
/48cfe424-adb5-49aa-8210-40733b92a7b4/images/5ff10066-1557-45a6-83fa-
5390a726275f/0c54f326-185d-4139-94c5-a0216ff42b1d
/48cfe424-adb5-49aa-8210-40733b92a7b4/images/372ea7eb-266d-41c7-a965-
96ce38e390e0/0f5d427f-57f4-49db-ad11-31ed538d92d3
/48cfe424-adb5-49aa-8210-40733b92a7b4/dom_md/ids
/48cfe424-adb5-49aa-8210-40733b92a7b4/images/64802852-597d-4354-b216-
0283d6164e21/2bf0504b-51d2-4f17-9e73-5ff392544f55
/48cfe424-adb5-49aa-8210-40733b92a7b4/dom_md/leases
/48cfe424-adb5-49aa-8210-40733b92a7b4/dom_md
/48cfe424-adb5-49aa-8210-40733b92a7b4/dom_md/metadata
```

Power on the ra-rhs-srv3 host. The system runs a self-healing daemon and this daemon starts healing the files between the storage nodes when both nodes are up. View the healing progress on the ra-rhs-srv4 host.

```
# gluster volume heal ROSESdatadomain info

Gathering list of entries to be healed on volume ROSESdatadomain has been
successful

Brick ra-rhs-srv3-
10g.storage.se.priv:/.rhs/ROSESdatadomain/ROSESdatadomain_brick
Number of entries: 0

Brick ra-rhs-srv4-
10g.storage.se.priv:/.rhs/ROSESdatadomain/ROSESdatadomain_brick
Number of entries: 4
/48cfe424-adb5-49aa-8210-40733b92a7b4/images/5ff10066-1557-45a6-83fa-
5390a726275f/0c54f326-185d-4139-94c5-a0216ff42b1d
/48cfe424-adb5-49aa-8210-40733b92a7b4/images/372ea7eb-266d-41c7-a965-
96ce38e390e0/0f5d427f-57f4-49db-ad11-31ed538d92d3
/48cfe424-adb5-49aa-8210-40733b92a7b4/images/64802852-597d-4354-b216-
0283d6164e21/2bf0504b-51d2-4f17-9e73-5ff392544f55
/48cfe424-adb5-49aa-8210-40733b92a7b4/dom_md/leases
```



If the self-healing process fails to start, a full healing process can be initiated using the **full** option instead of the **info** option.

```
# gluster volume heal ROSESdatadomain full
```

```
Launching heal operation to perform full self heal on volume ROSESdatadomain  
has been successful
```

```
Use heal info commands to check status
```

Wait 5 minutes to allow the Red Hat Storage nodes to heal the data bricks. This should be long enough for the small amount of data stored on them in this reference architecture.

Power off ra-rhs-srv4 and verify the applications still function correctly.

```
# curl -s http://earth-planet.example.org http://mars-planet.example.org  
http://neptune-planet.example.org http://jupiter-planet.example.org | grep  
title
```

```
<title>Welcome to Planet earth</title>  
<title>Welcome to Planet mars</title>  
<title>Welcome to Planet Neptune</title>  
<title>Welcome to Planet Jupiter</title>
```



13 Conclusion

Running the OpenShift Enterprise platform in a Red Hat Enterprise Virtualization environment using Red Hat Storage Server for the back end storage provides an excellent means for a highly available environment.

This reference architecture demonstrated OpenShift Enterprise node hosts running in a virtual machine can be easily failed over to a new virtual machine easily and quickly using the features available in Red Hat Enterprise Virtualization. With careful planning of the node host and using shared storage, data loss is easily prevented.

The functionality of the OpenShift Enterprise environment was demonstrated by creating a new application and verifying the new and existing applications continued to function as desired after a node host failure.

Data is further protected using replicated volumes in Red Hat Storage Server. This was demonstrated by failing each of the Red Hat Storage Server nodes and verifying the OpenShift Enterprise environment continued to function.



Appendix A: Resources

Red Hat Enterprise Virtualization Documentation

https://access.redhat.com/site/documentation/Red_Hat_Enterprise_Virtualization

Red Hat Storage 2.1 Installation and Configuration Guide

https://access.redhat.com/site/documentation/en-US/Red_Hat_Storage/2.1/html-single/Installation_and_Configuration_Guide/index.html

Red Hat Storage 2.1 Administration Guide

https://access.redhat.com/site/documentation/en-US/Red_Hat_Storage/2.1/html-single/Administration_Guide/index.html

Red Hat Storage 2.0 - Chapter 2 - Enabling Red Hat Storage in Red Hat Enterprise Virtualization Manager

https://access.redhat.com/site/documentation/en-US/Red_Hat_Storage/2.0/html-single/Quick_Start_Guide/index.html#chap-Quick_Start_Guide-vminrhs

OpenShift Enterprise Deployment Guide

https://access.redhat.com/site/documentation/en-US/OpenShift_Enterprise/1/html-single/Deployment_Guide/index.html

OpenShift Enterprise Administration Guide

https://access.redhat.com/site/documentation/en-US/OpenShift_Enterprise/1/html-single/Administration_Guide/index.html

OpenShift Enterprise Users Guide

https://access.redhat.com/site/documentation/en-US/OpenShift_Enterprise/1/html-single/User_Guide/index.html

OpenShift Enterprise Troubleshooting Guide

https://access.redhat.com/site/documentation/en-US/OpenShift_Enterprise/1/html-single/Troubleshooting_Guide/index.html

Red Hat Enterprise Linux 6.4 Security-Enhanced Linux User Guide

https://access.redhat.com/site/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Security-Enhanced_Linux/

OpenShift Enterprise Deploying and Managing a Private PaaS with OpenShift Enterprise

<http://www.redhat.com/resourcelibrary/reference-architectures/deploying-and-managing-a-private-paas-with-openshift-enterprise>



Appendix B: Scripts

This section contains the scripts used to setup and configure the environment used for this reference architecture. These scripts are provided as examples of automating the tasks involved in configuring this environment. These scripts and any scripts displayed in this paper are not meant to be used in a production environment, they do not contain the error logic required for a production environment. The scripts are provided as a learning tool only.

The scripts are executed on their respective host in the order they are listed unless otherwise specified. Most scripts require a base installation of Red Hat Enterprise Linux 6 before they can be executed. A brief explanation of each script is provided prior to the script.

These scripts are available for download from the Red Hat Customer Portal¹⁶.

The link to download the scripts in this appendix and this reference architecture is:

<https://access.redhat.com/site/node/541893/40>.

The direct link to download the scripts in a compressed tar format is:

<https://access.redhat.com/site/node/541893/40/1>

B.1 roses-common.sh

This script contains functions and variable definitions that are common to the remaining scripts. This script is not intended to be executed, it is sourced into the other scripts. It must be present in the same directory as the other scripts.

```
#!/bin/bash

# roses-common.sh

## Functions common to all
#

output() {
    echo -e "-- $@"
}

conf_networking() {

    output "Setting the hostname."
    sed -i -e "s/^\(HOSTNAME=\).*\/\1${host}.${domain}/" -e "t" -e "$
a\HOSTNAME=${host}.${domain}" /etc/sysconfig/network
    sed -i -e "s/^\(GATEWAY=\).*\/\1${gateway}/" -e "t" -e "$ a\GATEWAY=$
{gateway}" /etc/sysconfig/network

    hostname ${host}.${domain}

    output "Configuring name resolution."
    (
```

¹⁶ <http://access.redhat.com>



```
echo "search ${domain}"
for ns in ${nameservers}
do
    echo "nameserver ${ns}"
done
) > /etc/resolv.conf

output "Configuring the nics."
for nic in "${nics[@]}"
do
output "---- ${nic}"
    read dev ip mask <<< ${nic}
cat <<EOF > /etc/sysconfig/network-scripts/ifcfg-${dev}
DEVICE=${dev}
BOOTPROTO=none
ONBOOT=yes
IPADDR=${ip}
NETMASK=${mask}
EOF

    ifup ${dev}
    sleep 3
done

}

conf_updates() {
    rhn_reg_options=$1

    output "Registering the system with RHN Classic."
    rhnreg_ks ${rhn_reg_options} --username ${rhn_user} --password $
{rhn_passwd}

    output "Adding Channels."
    for channel in "${channels[@]}"
    do
        rhn-channel --user ${rhn_user} --password ${rhn_passwd} -a -c ${channel}
    done
}

conf_ntpd() {
    output "The date and time of this host is: $(date)"
    output "Enter the correct date and time or press <ENTER> to leave
unchanged (MMDDhhmmYYYY):"
    read new_date
    date ${new_date}

    output "Configuring the ntpd service to start on boot."
    chkconfig ntpd on
}
```



```
## Functions related to nameserver
#

conf_named() {
    output "Installing packages for named."
    yum -y install bind bind-utils

    output "Configuring /etc/named.conf"

    printf -v forwarders "%s; " "${named_forwarders[@]}"

cat <<EOF > /etc/named.conf
options {
    listen-on port 53 { any; };
    directory      "/var/named";
    dump-file      "/var/named/data/cache_dump.db";
    statistics-file "/var/named/data/named_stats.txt";
    memstatistics-file "/var/named/data/named_mem_stats.txt";
    allow-query    { any; };
    recursion yes;
    forwarders { ${forwarders} };

    dnssec-enable no;
    dnssec-validation no;
    dnssec-lookaside auto;

    /* Path to ISC DLV key */
    bindkeys-file "/etc/named.iscdlv.key";

    managed-keys-directory "/var/named/dynamic";
};

logging {
    channel default_debug {
        file "data/named.run";
        severity dynamic;
    };
};

zone "." IN {
    type hint;
    file "named.ca";
};

include "/etc/named.rfc1912.zones";
include "/etc/named.root.key";
EOF

    chkconfig named on
    lokkit --service dns
}

ose_check_sources() {
```



```
output "Checking repositories"
wget \
  https://github.com/openshift/openshift-extras/raw/enterprise-
2.0/admin/yum-validator/oo-admin-yum-validator \
  https://github.com/openshift/openshift-extras/raw/enterprise-
2.0/admin/yum-validator/etc/repos.ini \
  https://github.com/openshift/openshift-extras/raw/enterprise-
2.0/admin/yum-validator/etc/beta2.ini

mkdir yumvalidator
cd yumvalidator
wget \
  https://github.com/openshift/openshift-extras/raw/enterprise-
2.0/admin/yum-validator/yumvalidator/check_sources.py \
  https://github.com/openshift/openshift-extras/raw/enterprise-
2.0/admin/yum-validator/yumvalidator/repo_db.py \
  https://github.com/openshift/openshift-extras/raw/enterprise-
2.0/admin/yum-validator/yumvalidator/__init__.py

cd ..

chmod +x oo-admin-yum-validator

./oo-admin-yum-validator --repo-config repos.ini --report-all $@

output "\nAny indicated missing packages must be manually installed.
Install the packages and press <enter> to continue."
read ans

yum -y install yum-plugin-priorities

./oo-admin-yum-validator --fix-all --repo-config repos.ini $@

}
```

B.2 roses-ns.sh

This script is used to deploy the roses-ns nameserver. It should be the first script executed since name resolution is needed by other scripts.

```
#!/bin/bash

. roses-common.sh

# roses-ns.sh

## Functions
#

named_zone_def_roses() {
  output "Adding roses zone definition to /etc/named.conf"
```



```
pushd /var/named
echo 'include "/etc/named.roses.zones";' >> /etc/named.conf
zone_key_roses=$( dnssec-keygen -a hmac-md5 -b 256 -n USER example.org )
key_string=$( cat ${zone_key_roses}.private | sed -n -e 's/^Key:\s*//p' )
echo "${key_string}" > \$(cat ${host}.keyname

popd

cat <<EOF > /etc/named.roses.zones
zone "example.org." IN {
    type master;
    notify yes;
    file "dynamic/example.org.zone";
    allow-update { key example.org; };
};

zone "136.16.10.in-addr.arpa." IN {
    type master;
    notify yes;
    file "dynamic/136.16.10.in-addr.arpa.zone";
    allow-update { key example.org; };
};

key example.org {
    algorithm hmac-md5;
    secret "${key_string}";
};
EOF

chown -v root.named /etc/named.roses.zones
chcon --reference /etc/named.root.key /etc/named.roses.zones
}

named_zone_roses() {
    output "Creating zone files for roses."
}

cat <<'EOF' > /var/named/dynamic/example.org.zone
$TTL 86400
example.org.      IN SOA      roses-ns.example.org. root.roses-ns.example.org. (
                                2013082818
                                28800
                                7200
                                604800
                                86400
                                )
                                NS      roses-ns.example.org.

roses-ns          A      10.16.136.36
EOF
```



```
cat <<'EOF' > /var/named/dynamic/136.16.10.in-addr.arpa.zone
$TTL 86400
136.16.10.in-addr.arpa.      IN SOA      roses-ns.example.org. root.roses-ns.example.org. (
                             2013082802
                             28800
                             7200
                             604800
                             86400
                             )
                             NS      ns.roses.example.org.

36                            PTR    roses-ns.example.org.
EOF
```

```
chown -v named.named /var/named/dynamic/example.org.zone
chown -v named.named /var/named/dynamic/136.16.10.in-addr.arpa.zone
```

```
output "Starting named service."
service named start
```

```
(
echo "server 10.16.136.36"
```

```
for entry in "${ns_entries[@]}"
do
    read ip host <<< ${entry}
    echo "update add ${host}. 38400 A ${ip}"
done
```

```
echo "show"
echo "send"
```

```
for entry in "${ns_entries[@]}"
do
    read ip host <<< ${entry}
    rip=$( awk -F '.' '{printf "%s.%s.%s.%s", $4, $3, $2, $1;}' <<< ${ip} )
    echo "update add ${rip}.in-addr.arpa 38400 PTR ${host}."
done
```

```
echo "show"
echo "send"
) > /tmp/roses-updates.ns
```

```
output "Dynamically adding name resolution entries."
nsupdate -k /var/named/$1.key /tmp/roses-updates.ns
```

```
rndc freeze
rndc thaw
```

```
}
```



```
# -----

host="roses-ns"
domain="example.org"
nameservers="10.16.136.36 10.16.143.248"
gateway=10.16.143.254

nics=( \
    "eth0 10.16.136.36 255.255.248.0" \
)

rhn_user="admin"
rhn_passwd="CHANGEME"

channels=( \
)

named_forwarders=( \
    "10.16.143.248 port 53" \
    "10.16.143.247 port 53" \
)

ns_entries=( \
    "10.16.136.79 roses-rhelh.example.org" \
    "10.16.136.78 roses-rhevh.example.org" \
    "10.16.136.31 roses-rhevm.example.org" \
    "10.16.136.76 roses-broker.example.org" \
    "10.16.136.74 roses-node.example.org" \
)

# -----

(
# Check if a configuration file was passed on the command line.
# Source the file if there was.
#
if [ "$1" ]
then
    if [ -e $1 ]
    then
        . $1
    else
        output "Could not read input file $1."
        exit 1
    fi
fi

conf_networking
```



```

conf_updates

output "Updating the system packages."
yum -y update

conf_ntpd

conf_named

zone_key_roses=""
named_zone_def_roses

named_zone_roses ${zone_key_roses}
) | tee roses-ns.log

```

B.3 roses-rhss.sh

This script configures the Red Hat Storage Servers. It executes on a Red Hat Storage Server and not a Red Hat Enterprise Linux server. This script takes either the *roses-rhs-srv3.cfg* or *roses-rhs-srv4.cfg* file name as an option to indicate which storage server in being configured.

```

#!/bin/bash

. roses-common.sh

# roses-rhss.sh

## Functions
#

fs_prepare() {
    output "Creating logical volume"
    pvcreate --dataalignment ${pv_align} ${pv_device}
    vgcreate --physicalextentsize ${vg_extsz} ${vg_name} ${pv_device}
    lvcreate --size ${lv_size} --name ${lv_name} ${vg_name}

    output "Formatting logical volume as XFS."
    mkfs -t xfs -i size=512,maxpct=0 -d su=${raid_stripesz},sw=${raid_stripewidth},agcount=300 /dev/${vg_name}/${lv_name}

    mkdir -p /.rhs/${rhss_brickname}

    sed -i -e '$ a\dev/ROSESvg/datadomain_lv    /.rhs/ROSESdatadomain  xfs
inode64,logbsize=256k,noatime,nodiratime 1 3' /etc/fstab

    mount /.rhs/${rhss_brickname}

    mkdir /.rhs/${rhss_brickname}/${rhss_brickname}_brick
}

gluster_config() {

```



```
output "Configuring bricks."
gluster peer probe ${gluster_peer}

gluster volume create ${rhss_brickname} replica 2 ${gluster_local}:/rhs/${rhss_brickname}/${rhss_brickname}_brick ${gluster_peer}:/rhs/${rhss_brickname}/${rhss_brickname}_brick
}

rhss_tune_fw() {
gluster volume set ${rhss_brickname} group virt
gluster volume set ${rhss_brickname} storage.owner-uid 36
gluster volume set ${rhss_brickname} storage.owner-gid 36
gluster volume start ${rhss_brickname}

output "Configuring firewall."
iptables --append INPUT --protocol tcp --match state --state NEW --match tcp --dport 111 --jump ACCEPT

iptables --append INPUT --protocol udp --match state --state NEW --match udp --dport 111 --jump ACCEPT

iptables --append INPUT --protocol tcp --match state --state NEW --match tcp --dport 24007 --jump ACCEPT

iptables --append INPUT --protocol tcp --match state --state NEW --match tcp --dport ${gluster_brick_ports} --jump ACCEPT

service iptables save

output "Tuning server."
tuned-adm profile rhs-virtualization
tuned-adm active
}

# -----

host=""
domain="example.org"
nameservers="10.16.143.247 10.16.143.248"
gateway=10.16.143.254

nics=( \
)

rhel_user="admin"
rhel_passwd="CHANGEME"

channels=( \
"rhel-x86_64-server-6-rhs-2.1" \
"rhel-x86_64-server-sfs-6.4.z" \
```




```
)

pv_device=/dev/sdb
pv_align="1024KB"
vg_extsz="64M"
vg_name=ROSESvg
lv_size=1T
lv_name=datadomain_lv

raid_stripesize=256k
raid_stripewidth=4

rhss_brickname=ROSESdatadomain

gluster_local=ra-rhs-srv3-10g.storage.se.priv
gluster_peer=ra-rhs-srv4-10g.storage.se.priv

gluster_brick_ports="24009:24016"

# -----

(
# Check if a configuration file was passed on the command line.
# Source the file if there was.
#
if [ "$1" ]
then
    if [ -e $1 ]
    then
        . $1
    else
        output "Could not read input file $1."
        exit 1
    fi
fi

conf_networking
conf_updates "--use-eus-channel"

output "Updating the system packages."
yum -y update

conf_ntpd

fs_prepare

if [ ! "${donot_config_gluster}" ]
then
    gluster_config
fi

rhss_tune_fw
) | tee roses-rhss.log
```



B.4 roses-rhs-srv3.cfg

This is the configuration files used to install and configure the ra-rhs-srv3 storage server.

```
# roses-rhs-srv3.cfg

# -----

host="ra-rhs-srv3"

nics=( \
  "em1  10.16.143.95  255.255.248.0" \
  "p1p1 172.31.143.95 255.255.0.0" \
)
```

B.5 roses-rhs-srv4.cfg

This is the configuration files used to install and configure the ra-rhs-srv4 storage server.

```
# roses-rhs-srv4.cfg

# -----

host="ra-rhs-srv4"

nics=( \
  "em1  10.16.143.96  255.255.248.0" \
  "p1p1 172.31.143.96 255.255.0.0" \
)

donot_config_gluster="OnlyDoneOnOneNode"
```

B.6 roses-rhelh.sh

This script is used to install and configure the roses-rhelh hypervisor.

```
#!/bin/bash

. roses-common.sh

# roses-rhelh.sh

## Functions
#

conf_firewall() {

  output "Configuring the firewall"
```



```
cat <<'EOF' > /etc/sysconfig/iptables
# oVirt default firewall configuration. Automatically generated by vdsmd bootstrap script.
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT

-A INPUT -i lo -j ACCEPT
# vdsmd
-A INPUT -p tcp --dport 54321 -j ACCEPT
# SSH
-A INPUT -p tcp --dport 22 -j ACCEPT
# snmp
-A INPUT -p udp --dport 161 -j ACCEPT

# libvirt tls
-A INPUT -p tcp --dport 16514 -j ACCEPT

# guest consoles
-A INPUT -p tcp -m multiport --dports 5634:6166 -j ACCEPT

# migration
-A INPUT -p tcp -m multiport --dports 49152:49216 -j ACCEPT

# Reject any other input traffic
-A INPUT -j REJECT --reject-with icmp-host-prohibited
-A FORWARD -m physdev ! --physdev-is-bridged -j REJECT --reject-with icmp-host-prohibited
COMMIT
EOF

service iptables restart
}

# -----

host="roses-rhelh"
domain="example.org"
nameservers="10.16.136.36"
gateway=10.16.143.254

nics=( \
    "eth0 10.16.136.79 255.255.248.0" \
)

rhn_user="admin"
rhn_passwd="CHANGEME"

channels=( \
    "rhel-x86_64-rhev-mgmt-agent-6" \
```



```
"rhel-x86_64-server-rhscclient-6" \  
)  
  
# -----  
  
(  
# Check if a configuration file was passed on the command line.  
# Source the file if there was.  
#  
if [ "$1" ]  
then  
    if [ -e $1 ]  
    then  
        . $1  
    else  
        output "Could not read input file $1."  
        exit 1  
    fi  
fi  
  
conf_networking  
conf_updates  
  
output "Updating the system packages."  
yum -y update  
  
yum -y install glusterfs-fuse  
  
conf_ntpd  
conf_firewall  
) | tee roses-rhelh.log
```

B.7 roses-rhevm.sh

This script configures the roses-rhevm host. It installs the rhevm software and configures the data center, cluster, networks, and storage. It also adds the hypervisors to the environment and creates the virtual machines.

This script must be run after the nameserver, storage servers, and hypervisors are configured.

```
#!/bin/bash  
  
. roses-common.sh  
  
# roses-rhevm.sh  
  
## Functions  
#  
  
conf_rhevm() {
```



```
output "Installing and configuring the Red Hat Enterprise Virtualization Manager"
```

```
yum -y install rhvm rhvm-doc  
rhvm-setup --gen-answer-file rhvm-answer.txt
```

```
sed -i \  
-e "s/^(AUTH_PASS=).*\1${rhvm_passwd}/" \  
-e "s/^(DB_LOCAL_PASS=).*\1${rhvm_db_passwd}/" \  
-e "s/^(CONFIG_NFS=).*\1no/" \  
-e "s/^(OVERRIDE_FIREWALL=).*\1iptables/" \  
rhvm-answer.txt
```

```
rhvm-setup --answer-file rhvm-answer.txt
```

```
sleep 5  
}
```

```
rhvm-command() {
```

```
# The ^[ in the sed statement is an escape character. It is place into the editor by  
# pressing CTRL-v and then pressing the escape key.  
rhvm-shell --ca-file ${host}.cacer <<< "$1" | sed -e "/#/, /^\$/d" -e "/#/,/(connected)/d"
```

```
}
```

```
conf_rhev_conn() {
```

```
output "Retrieving CA Cert."  
wget -O ${host}.cacer http://${host}/ca.crt
```

```
output "Creating /root/.rhevshellrc"  
cat <<EOF >/root/.rhevshellrc
```

```
[cli]
```

```
autopage = True
```

```
[ovirt-shell]
```

```
username = admin@internal
```

```
url = https://roses-rhev/api
```

```
insecure = False
```

```
filter = False
```

```
session_timeout = -1
```

```
timeout = -1
```

```
dont_validate_cert_chain = False
```

```
password = ${rhvm_passwd}
```

```
EOF
```

```
}
```

```
chk_rhev() {
```

```
output "Checking if environment contains Red Hat Enterprise Virtualization Hypervisors"
```

```
for entry in "${Ahosts[@]}"
```

```
do
```

```
read t_hyper t_ip t_pass t_cl t_type t_trash <<< ${entry}
```

```
if [ "${t_type}" = "rhev" ]
```



```
then
    t_rhev[${#t_rhev[@]}]="${t_hyper} - ${t_ip}"
fi
done

if [ "${t_rhev}" ]
then
    output "Log into the following Red Hat Enterprise Virtualization Hypervisors"
    output "using the admin account and register them to this host: ${host}\n"

    for t_hyp in "${t_rhev[@]}"
    do
        output "${t_hyp}"
    done

    output "\nPress <ENTER> once all the Red Hat Enterprise Virtualization Hypervisors have been
registered."
    read ans
fi
}

rhev_cr_dc() {
    for entry in "${Adatacenters[@]}"
    do
        output "Creating Data Center - ${entry}"
        read t_dc t_stype t_maj t_min <<< ${entry}
        rhvm-command "add datacenter --name \"${t_dc}\" --storage_type ${t_stype} --version-major $
${t_maj} --version-minor ${t_min} --expect"
    done
}

rhev_cr_cl() {
    for entry in "${Aclusters[@]}"
    do
        output "Creating Cluster - ${entry}"
        read t_cl t_dc t_maj t_min t_virt t_cpu <<< ${entry}
        rhvm-command "add cluster --data_center-name \"${t_dc}\" --name \"${t_cl}\" --version-major $
${t_maj} --version-minor ${t_min} --cpu-id \"${t_cpu}\" --virt_service ${t_virt} --gluster_service false
--expect"
    done
}

rhev_cr_net() {
    for entry in "${Anetworks[@]}"
    do
        output "Creating Network - ${entry}"
        read t_net t_dc t_cl t_dis <<< ${entry}
        rhvm-command "add network --name \"${t_net}\" --data_center-name \"${t_dc}\" --display ${t_dis}"
        rhvm-command "add network --cluster-identifier \"${t_cl}\" --name \"${t_net}\""
    done
}
```



```
}

rhevm_hyp_add() {
  for entry in "${Ahosts[@]}"
  do
    output "Adding and Approving hypervisors"
    read t_hyper t_ip t_pass t_cl t_type t_trash <<< ${entry}
    if [ "${t_type}" = "rhev" ]
    then
      output "Approving ${t_hyper}"
      rhevm-command "action host \"${t_hyper}\" approve --cluster-name \"${t_cl}\""
    else
      output "Adding ${t_hyper}"
      rhevm-command "add host --name \"${t_hyper}\" --address \"${t_ip}\" --root_password \"${t_pass}\"
--cluster-name \"${t_cl}\""
    fi
  done
}

rhevm_hyp_stats() {
  if [ -e /tmp/waiting_hypervisors ]
  then
    rm -f /tmp/waiting_hypervisors
  fi

  while [ ! -e /tmp/waiting_hypervisors ]
  do
    unset wait_host
    for entry in "${Ahosts[@]}"
    do
      read t_hyper t_ip t_pass t_cl t_type t_trash <<< ${entry}
      state=$( rhevm-command "show host \"${t_hyper}\"" | sed -n 's/^status-state.*: //p' )

      output " -- ${t_hyper} is ${state}"

      if [ "${state}" != "up" ]
      then
        wait_host=y
      fi
    done

    if [ "${wait_host}" ]
    then
      output " Sleeping 60 seconds"
      sleep 60
    else
      > /tmp/waiting_hypervisors
    fi
  done
}
```



```
rhevm_conf_pm() {
# Power Management
  for entry in "${Ahosts[@]}"
  do
#   read name ip passwd cl hyp pm pmttype pmip pmuser pmpass<<< ${entry}
    read t_hyper t_ip t_pass t_cl t_type t_pm t_pmttype t_pmip t_pmuser t_pmpass <<< ${entry}

    output "Configuring Power Management - ${t_hyper}:${t_pmttype}"

    rhevm-command "update host \"${t_hyper}\" --power_management-enabled ${t_pm}
--power_management-type \"${t_pmttype}\" --power_management-address \"${t_pmip}\"
--power_management-username \"${t_pmuser}\" --power_management-password \"${t_pmpass}\""
    done
  }

rhevm_attach_net() {
# Attach networks to hypervisors
  for entry in "${Ahnics[@]}"
  do
#   read name network host ip netmask <<< ${entry}
    read t_int t_net t_hyper t_ip t_mask <<< ${entry}

    output "Attaching network ${t_net} to hypervisor - ${t_hyper}:${t_int}"

    rhevm-command "action nic \"${t_int}\" attach --host-identifier \"${t_hyper}\" --network-name \"${t_net}\""

    rhevm-command "action host \"${t_hyper}\" deactivate"

    output " -- Configuring network - ${t_ip}"
    rhevm-command "update nic \"${t_int}\" --host-identifier \"${t_hyper}\" --boot_protocol static --ip-address \"${t_ip}\" --ip-netmask \"${t_mask}\""

    rhevm-command "action host \"${t_hyper}\" activate"

    rhevm-command "action host \"${t_hyper}\" commitnetconfig"
  done
}

rhevm_cr_sd() {
# Create Storage Domains
  for entry in "${Astoragedomains[@]}"
  do
    read t_sd t_dc t_hyper t_dom t_stype t_spath t_vfs <<< ${entry}

    output "Creating Storage Domain - ${t_sd} - ${t_spath}"

    rhevm-command "add storagedomain --name \"${t_sd}\" --host-name \"${t_hyper}\" --type ${t_dom}
--storage-type \"${t_stype}\" --storage-path \"${t_spath}\" --storage-vfs_type \"${t_vfs}\" --expect"

    rhevm-command "add storagedomain --name \"${t_sd}\" --datacenter-identifier \"${t_dc}\" --expect"
  done
}
```




```
}

rhevm_cr_vm() {
# Create VMS
for entry in "${Avirtualmachines[@]}"
do
    read t_vm t_tplt t_mem t_cpu t_os t_type t_ha t_cl t_power <<< ${entry}

    output "Creating VM - ${t_vm}"

    rhevm-command "add vm --name \"${t_vm}\" --template-name \"${t_tplt}\" --cluster-name \"${t_cl}\"
--memory ${t_mem} --cpu-topology-cores ${t_cpu} --os-type \"${t_os}\" --type ${t_type}
--high_availability-enabled ${t_ha} --expect"

done
}

rhevm_cr_disk() {
# Create Disks
for entry in "${Adisks[@]}"
do
    read t_disk t_size t_drv t_fmt t_boot t_share t_sd t_vm <<< ${entry}

    if [ ${t_vm} ]
    then
        output "Creating disk - ${t_disk} on ${t_vm}"

        rhevm-command "add disk --name \"${t_disk}\" --provisioned_size ${t_size} --interface ${t_drv}
--format ${t_fmt} --bootable ${t_boot} --shareable ${t_share} --storage_domains-
storage_domain \"storage_domain.name=${t_sd}\" --vm-identifier \"${t_vm}\" --expect"
    else
        output "Creating disk - ${t_disk}"

        rhevm-command "add disk --name \"${t_disk}\" --provisioned_size ${t_size} --interface ${t_drv}
--format ${t_fmt} --bootable ${t_boot} --shareable ${t_share} --storage_domains-
storage_domain \"storage_domain.name=${t_sd}\" --expect"
    fi

done
}

rhevm_cr_vmnics() {
# Create Nics
for entry in "${Avmnics[@]}"
do
    read name mac net interface link plug vm <<< ${entry}
    read t_nic t_mac t_net t_drv t_link t_plug t_vm <<< ${entry}

    output "Creating NIC - ${t_nic}:${t_mac} on ${t_vm}"

    rhevm-command "add nic --name \"${t_nic}\" --network-name \"${t_net}\" --mac-address \"${t_mac}\""
```



```
--interface ${t_drv} --linked ${t_link} --plugged ${t_plug} --expect --vm-identifier \"${t_vm}\"""

done
}

# -----

host="roses-rhev"
domain="example.org"
nameservers="10.16.136.36"
gateway=10.16.143.254

nics=( \
    "eth0 10.16.136.31 255.255.248.0" \
)

rhn_user="admin"
rhn_passwd="CHANGE"

channels=( \
    "rhel-x86_64-server-supplementary-6" \
    "rhel-x86_64-server-6-rhev-3.2" \
    "jbappplatform-6-x86_64-server-6-rpm" \
)

rhev_passwd="CHANGE"
rhev_db_passwd="CHANGE"

# Name storage_type version-major version-minor
Adatacenters=( \
    "DC-Main posixfs 3 2" \
)

# Name data_center version-major version-minor en_virt cpu-id
Aclusters=( \
    "CL-Production DC-Main 3 2 true Intel Nehalem Family" \
)

# Name data_center cluster display_net
Anetworks=( \
    "RHStorage DC-Main CL-Production false" \
)

# Name IP passwd cluster hypervisor power_management pm_type pm_ip pm_user pm_passwd
Ahosts=( \
    "roses-rhev.example.org 10.16.136.78 CHANGE CL-Production rhv true ilo2 10.16.143.235
Administrator 24^goldA" \
    "roses-rhel.example.org 10.16.136.79 CHANGE CL-Production rhel true ilo2 10.16.143.238
Administrator 24^goldA" \
)
```



```
# Name data_center host domain type storage-type storage-path vfs-type
Astoragedomains=( \
  "Data-DC-Main DC-Main roses-rhelh.example.org data posixfs ra-rhs-srv3-
10g.storage.se.priv:ROSESdatadomain glusterfs" \
)

# Name prov-size interface format bootable shareable storagedomain vm
Adisks=( \
  "roses-broker_root 12884901888 virtio cow true false Data-DC-Main roses-broker" \
  "roses-node_root 12884901888 virtio cow true false Data-DC-Main roses-node" \
  "roses-backup-node_root 12884901888 virtio cow true false Data-DC-Main roses-backup-node" \
  "roses-gears_share 21474836480 virtio raw false true Data-DC-Main" \
)

# Name mac network interface linked plugged vm
Avmnics=( \
  "eth0 00:1a:4a:a8:6e:3f rhevm virtio true true roses-broker" \
  "eth0 00:1a:4a:a8:6e:3a rhevm virtio true true roses-node" \
  "eth0 00:1a:4a:a8:6e:3b rhevm virtio true true roses-backup-node" \
)

# Name network host ip netmask
Ahnics=( \
  "eth1 RHStorage roses-rhevh.example.org 172.31.136.79 255.255.0.0" \
  "eth1 RHStorage roses-rhelh.example.org 172.31.136.78 255.255.0.0" \
)

# Name template memory cpus os type ha cluster power_on
Avirtualmachines=( \
  "roses-broker Blank 1073741824 1 rhel_6x64 server true CL-Production true" \
  "roses-node Blank 1073741824 1 rhel_6x64 server true CL-Production true" \
  "roses-backup-node Blank 1073741824 1 rhel_6x64 server true CL-Production true" \
)

# -----

(
# Check if a configuration file was passed on the command line.
# Source the file if there was.
#
if [ "$1" ]
then
  if [ -e $1 ]
  then
    . $1
  else
    output "Could not read input file $1."
    exit 1
  fi
fi

conf_networking
conf_updates
```



```
output "Updating the system packages."
yum -y update
```

```
conf_ntpd
```

```
conf_rhevm
```

```
conf_rhev_conn
chk_rhev
```

```
rhevm_cr_dc
rhevm_cr_cl
rhevm_cr_net
```

```
rhevm_hyp_add
rhevm_attach_net
rhevm_hyp_stats
rhevm_conf_pm
```

```
rhevm_cr_sd
```

```
rhevm_cr_vm
rhevm_cr_disk
rhevm_cr_vmnic
) | tee roses-rhevm.log
```

B.8 roses-broker.sh

This script configures the roses-broker host. It must be the executed before the remaining scripts.

```
#!/bin/bash

. roses-common.sh

# roses-broker.sh

## Functions
#

rhevm_guest_agent() {

    output "Installing the Red Hat Enterprise Virtualiation Guest Agent"
    yum -y install rhevm-guest-agent

    service ovirt-guest-agent start
    chkconfig ovirt-guest-agent on
}

broker_mongodb() {
```



```
output "Configuring the MongoDB"
yum -y install mongodb-server
```

```
sed -i -e "s/^(#*)auth\s*=.*\/auth = true/" -e "/^smallfiles .*$/d" /etc/mongodb.conf
```

```
chkconfig mongod on
service mongod start
```

```
sleep 40
```

```
output "Adding MongoDB users"
```

```
cat <<EOF | mongo
use admin
db.addUser("admin", "${mdbadmin_passwd}")
db.auth("admin", "${mdbadmin_passwd}")
use openshift_broker
db.addUser("openshift", "${moseuser_passwd}")
EOF
}
```

```
broker_activemq() {
```

```
    output "Configuring ActiveMQ"
```

```
    yum -y install activemq activemq-client
```

```
    wget https://raw.githubusercontent.com/openshift/openshift-extras/enterprise-1.2/enterprise/install-
scripts/activemq.xml -O /etc/activemq/activemq.xml
```

```
    sed -i \
    -e "s/^(brokerName=\\\")activemq.example.com/\\1${host}.${domain}/" \
    -e "s/^(username=\\\"mcollective\\\" password=\\\")marionette/\\1${mcollective_passwd}/" \
    -e "s/^(username=\\\"admin\\\" password=\\\")secret/\\1${activemq_passwd}/" \
    /etc/activemq/activemq.xml
```

```
    sed -i -e "/name=\\\"authenticate\\\"/s/false/true/" /etc/activemq/jetty.xml
```

```
    sed -i -e "/name=\\\"port\\\"/a<property name=\\\"host\\\" value=\\\"127.0.0.1\\\" />" /etc/activemq/jetty.xml
```

```
    sed -i -e "/admin:/s/admin,/${activemq_passwd},/" /etc/activemq/jetty-realm.properties
```

```
    chkconfig activemq on
    lokkit --port 61613:tcp
}
```

```
broker_mcollective() {
```

```
    output "Configuring MCollective"
```

```
    yum -y install mcollective-client
```

```
    cat <<EOF > /etc/mcollective/client.cfg
```



```
topicprefix = /topic/
main_collective = mcollective
collectives = mcollective
libdir = /opt/rh/ruby193/root/usr/libexec/mcollective
logfile = /var/log/mcollective-client.log
loglevel = debug
direct_addressing = 1

# Plugins
securityprovider = psk
plugin.psk = ${mc_psk}

connector = activemq
plugin.activemq.pool.size = 1
plugin.activemq.pool.1.host = ${host}.${domain}
plugin.activemq.pool.1.port = 61613
plugin.activemq.pool.1.user = mcollective
plugin.activemq.pool.1.password = ${mcollective_passwd}

# Facts
factssource = yaml
plugin.yaml = /etc/mcollective/facts.yaml
EOF
}

broker_broker() {

    output "Configuring the broker application"
    yum -y install openshift-origin-broker openshift-origin-broker-util rubygem-openshift-origin-auth-remote-
user rubygem-openshift-origin-msg-broker-mcollective rubygem-openshift-origin-dns-nsupdate

    sed -i "/VirtualHost/,/VirtualHost/ d" /etc/httpd/conf.d/ssl.conf

    touch /var/log/mcollective-client.log
    chown apache:root /var/log/mcollective-client.log

    chkconfig httpd on

    lokkit --service=https
    lokkit --service=http

    sed -i -e "s/ServerName .*/ServerName `hostname`/"
/etc/httpd/conf.d/000002_openshift_origin_broker_servername.conf

    output " Generating keys and salts"

    openssl genrsa -out /etc/openshift/server_priv.pem 2048

    openssl rsa -in /etc/openshift/server_priv.pem -pubout > /etc/openshift/server_pub.pem

    auth_salt=$( openssl rand -base64 64 | tr -d '\n' )
    session_secret=$( openssl rand -hex 64 | tr -d '\n' )
```



```
sed -i -e "/AUTH_SALT/cAUTH_SALT=\"\${auth_salt}\"\" \" \
-e "/SESSION_SECRET/cSESSION_SECRET=\"\${session_secret}\"\" \" \
/etc/openshift/broker.conf

mkdir /root/.ssh
chown root.root /root/.ssh
chmod 700 /root/.ssh

ssh-keygen -t rsa -b 2048 -N "" -f /root/.ssh/rsync_id_rsa

output " Configuring selinux"
setsebool -P httpd_unified=on httpd_execmem=on httpd_can_network_connect=on
httpd_can_network_relay=on httpd_run_stickshift=on named_write_master_zones=on allow_ybind=on

fixfiles -R ruby193-rubygem-passenger restore
fixfiles -R ruby193-mod_passenger restore
restorecon -rv /var/run
restorecon -rv /opt

output " Configuring broker.conf"

sed -i -e "s/^CLOUD_DOMAIN=.*$/CLOUD_DOMAIN=${domain}/" /etc/openshift/broker.conf
sed -i "/MONGO_USER/cMONGO_USER=\"openshift\"\" /etc/openshift/broker.conf
sed -i "/MONGO_PASSWORD/cMONGO_PASSWORD=\"\${moseuser_passwd}\"\" \
/etc/openshift/broker.conf
sed -i "/MONGO_DB/cMONGO_DB=\"openshift_broker\"\" /etc/openshift/broker.conf

output " Configuring plugins"

cd /etc/openshift/plugins.d/
cp openshift-origin-auth-remote-user.conf.example openshift-origin-auth-remote-user.conf
cp openshift-origin-msg-broker-mcollective.conf.example openshift-origin-msg-broker-mcollective.conf

output "\nNeed to update key get key from name server."
output " Press <enter> when ready"
read ans
key_string=$( ssh roses-ns "cd /var/named; cat \"$(cat ${domain}.keyname).private" | sed -n -e
's/^Key:\s*//p' )

cat <<-EOF > /etc/openshift/plugins.d/openshift-origin-dns-nsupdate.conf
BIND_SERVER="10.16.136.36"
BIND_PORT=53
BIND_KEYNAME="${domain}"
BIND_KEYVALUE="${key_string}"
BIND_ZONE="${domain}"
EOF

cp /var/www/openshift/broker/httpd/conf.d/openshift-origin-auth-remote-user-basic.conf.sample
/var/www/openshift/broker/httpd/conf.d/openshift-origin-auth-remote-user.conf

htpasswd -b -c /etc/openshift/htpasswd ${ose_user} ${ose_passwd}

cd /var/www/openshift/broker
```



```
scl enable ruby193 'bundle --local'

chkconfig openshift-broker on
service httpd start
service openshift-broker start

}

# -----

host="roses-broker"
domain="example.org"
nameservers="10.16.136.36"
gateway=10.16.143.254

nics=( \
    "eth0 10.16.136.76 255.255.248.0" \
)

rhn_user="admin"
rhn_passwd="CHANGE_ME"

channels=( \
    "rhel-x86_64-server-6-ose-1.2-infrastructure" \
    "rhel-x86_64-server-6-ose-1.2-rhc" \
    "rhel-x86_64-rhev-agent-6-server" \
    "rhel-x86_64-server-6-rhscl-1" \
)

rhevm_passwd="CHANGE_ME"
rhevm_db_passwd="CHANGE_ME"

mdbadmin_passwd="CHANGE_ME"
moseuser_passwd="CHANGE_ME"

mcollective_passwd="CHANGE_ME"
activemq_passwd="CHANGE_ME"

mc_psk="thisispreshared"

ose_user=rosesuser
ose_passwd=CHANGE_ME

# -----

(
# Check if a configuration file was passed on the command line.
# Source the file if there was.
#
if [ "$1" ]
then
```




```
if [ -e $1 ]
then
. $1
else
output "Could not read input file $1."
exit 1
fi
fi

conf_networking
conf_updates

output "Updating the system packages."
yum -y update

conf_ntpd

rhev_guest_agent

ose_check_sources --role broker --role client

broker_mongodb
broker_activemq
broker_mcollective
broker_broker

init 6

) | tee roses-broker.log
```

B.9 roses-node.sh

This script configures the roses-node and roses-backup-node hosts. If no options are given, it configures the roses-node host. If the roses-backup-node.cfg file name is passed as an option to this script, the script configures the roses-backup-node host instead.

```
#!/bin/bash

. roses-common.sh

# roses-node.sh

## Functions
#

rhev_guest_agent() {

output "Installing the Red Hat Enterprise Virtualization Guest Agent"
yum -y install rhv-guest-agent

service ovirt-guest-agent start
chkconfig ovirt-guest-agent on
}
```



```
node_mcollective() {

    output "Configuring MCollective"

    yum -y install openshift-origin-msg-node-mcollective

    cat <<EOF > /etc/mcollective/server.cfg
topicprefix = /topic/
main_collective = mcollective
collectives = mcollective
libdir = /opt/rh/ruby193/root/usr/libexec/mcollective
logfile = /var/log/mcollective.log
loglevel = debug

daemonize = 1
direct_addressing = 1

# Plugins
securityprovider = psk
plugin.psk = ${mc_psk}

connector = activemq
plugin.activemq.pool.size = 1
plugin.activemq.pool.1.host = ${broker}.${domain}
plugin.activemq.pool.1.port = 61613
plugin.activemq.pool.1.user = mcollective
plugin.activemq.pool.1.password = ${mcollective_passwd}

# Facts
factssource = yaml
plugin.yaml = /etc/mcollective/facts.yaml
EOF

    chkconfig mcollective on
}

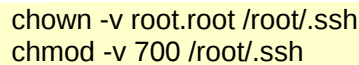
node_node() {

    output "Configuring the node"

    yum -y install \
        rubygem-openshift-origin-node \
        ruby193-rubygem-passenger-native \
        openshift-origin-port-proxy \
        openshift-origin-node-util

    yum -y install $( yum list "openshift-origin-cartridge*" | grep -o "openshift-origin-cartridge.\w*" )

    output " Configuring ssh"
    mkdir /root/.ssh
```



```
output "\n Need to add brokers public key to authorized_keys."
```

output " Press <enter> when ready to continue"

```
read ans
```

```
ssh root@${broker} cat /root/.ssh/rsync_id_rsa.pub >> /root/.ssh/authorized_keys
```

```
chmod -v 600 /root/.ssh/authorized_keys
```

```
cat <<EOF >> /etc/ssh/sshd_config
```

AcceptEnv GIT_SSH

MaxSessions 40

MaxStartups 40

EOF

output " Configuring services and firewall"

```
lokkit --service=https
```

```
lokkit --service=http
```

```
lokkit --port=8443:tcp
```

```
chkconfig httpd on
```

```
chkconfig oddjobd on
```

```
chkconfig openshift-node-web-proxy on
```

output " Configuring PAM and cgroups"

```
sed -i -e 's|pam_selinux|pam_openssh|g' /etc/pam.d/sshd
```

```
for file in "runuser" "runuser-l" "sshd" "su" "system-auth-ac"
```

do

```
temp="/etc/pam.d/${file}"
```

```
if ! grep -q "pam_namespace.so" "${temp}"
```

then

```
echo -e "session\t\t[default=1 success=ignore]\t\tpam_succeed_if.so quiet shell = /usr/bin/oo-trap-user" >> "${temp}"
```

```
echo -e "session\t\trequired\tppam namespace.so no unmount on close" >> "${temp}"
```

fi

```
if ! grep -q "pam_cgroup" "${temp}"
```

then

```
echo -e "session\t\toptional\tpam cgroup.so" >> "${temp}"
```

fi

done

```
echo "/tmp      \${HOME}/.tmp/      user:iscript=/usr/sbin/oo-namespace-init root,adm" >
```

```
/etc/security/namespace.d/tmp.conf
```

```
echo "/dev/shm tmpfs tmpfs:mntopts=size=5M:iscript=/usr/sbin/oo-namespace-init root,adm" >
```

```
/etc/security/namespace.d/shm.conf
```

```
restorecon -rv /etc/cgconfig.conf
```

```
mkdir -p /cgroup
```

```
restorecon -rv /cgroup
```

```
chkconfig cgconfig on
```

```
chkconfig cgred on
```



```
chkconfig openshift-cgroups on
}

node_conf_cgroups() {
    output " Configuring disk quotas"
    ose_mp=$( df -P /var/lib/openshift | grep -v "^Filesystem" | awk '{print $6}' )

    cp /etc/fstab{,.orig}
    awk '$2 ~ "^${ose_mp}$" { $4 = "usrquota," $4 } 1' /etc/fstab.orig > /etc/fstab
    mount -o remount ${ose_mp}
    quotacheck -cmug ${ose_mp}
    restorecon ${ose_mp}/aquota.user
}

node_conf_security() {
    output "Configure SELinux"
    setsebool -P httpd_unified=on httpd_can_network_connect=on httpd_can_network_relay=on
    httpd_read_user_content=on httpd_enable_homedirs=on httpd_run_stickshift=on
    allow_polyinstantiation=on

    restorecon -rv /var/run
    restorecon -rv /usr/sbin/mcollectived /var/log/mcollective.log /var/run/mcollectived.pid
    restorecon -rv /var/lib/openshift /etc/openshift/node.conf /etc/httpd/conf.d/openshift

    output "Tuning system"
    cat <<EOF >> /etc/sysctl.conf
kernel.sem = 250 32000 32 4096
net.ipv4.ip_local_port_range = 15000 35530
net.netfilter.nf_conntrack_max = 1048576
EOF
}

node_conf_misc() {
    output "Configure port proxy"
    lokkit --port=35531-65535:tcp
    chkconfig openshift-port-proxy on
    service openshift-port-proxy start
    chkconfig openshift-gears on

    output "Configure node settings"
    sed -i \
    -e "s/(PUBLIC_HOSTNAME=\\").*\\1${host}.${domain}\\1/" \
    -e "s/(PUBLIC_IP=\\").*\\1${pub_ip}\\1/" \
    -e "s/(BROKER_HOST=\\").*\\1${broker}\\1/" \
    -e "s/(CLOUD_DOMAIN=\\").*\\1${domain}\\1/" \
    /etc/openshift/node.conf

    echo ${broker} > /etc/openshift/env/OPENSHIFT_BROKER_HOST
    echo ${domain} > /etc/openshift/env/OPENSHIFT_CLOUD_DOMAIN

    sed -i -e "s/ServerName .*$/ServerName `hostname`/"
    /etc/httpd/conf.d/000001_openshift_origin_node.conf

    /etc/cron.minutely/openshift-facts
```



```
}

# -----

host="roses-node"
domain="example.org"
nameservers="10.16.136.36"
gateway=10.16.143.254

nics=( \
    "eth0 10.16.136.74 255.255.248.0" \
)

pub_ip="10.16.136.74"

rhn_user="admin"
rhn_passwd="CHANGEME"

channels=( \
    "rhel-x86_64-rhev-agent-6-server" \
    "rhel-x86_64-server-6-ose-1.2-node" \
    "rhel-x86_64-server-6-ose-1.2-jbosseap" \
    "jbappplatform-6-x86_64-server-6-rpm" \
    "jb-ews-2-x86_64-server-6-rpm" \
    "rhel-x86_64-server-6-rhscl-1" \
)

broker="roses-broker"

mcollective_passwd="CHANGEME"
mc_psk="thisispreshared"

# -----

(
# Check if a configuration file was passed on the command line.
# Source the file if there was.
#
if [ "$1" ]
then
    if [ -e $1 ]
    then
        . $1
    else
        output "Could not read input file $1."
        exit 1
    fi
fi

conf_networking
conf_updates
```



```
output "Updating the system packages."
yum -y update

conf_ntpd

rhevm_guest_agent

ose_check_sources --role node --role node-eap

node_mcollective
node_node
node_conf_cgroups
node_conf_security
node_conf_misc

init 6

) | tee roses-node.log
```

B.10 roses-backup-node.cfg

This configuration file is used to configure the roses-backup-node host. It redefines some of the functions and variables in the roses-node.sh script.

```
# roses-backup-node.cfg

node_mcollective() {
    output "Configuring MCollective"

    yum -y install openshift-origin-msg-node-mcollective
}

node_conf_misc() {
    output "Configure port proxy"
    lokkit --port=35531-65535:tcp

    sed -i -e "s/ServerName .*$/ServerName `hostname`/"
    /etc/httpd/conf.d/000001_openshift_origin_node.conf

    /etc/cron.minutely/openshift-facts

    for i in mcollective \
        httpd \
        ntpd \
        oddjobd \
        openshift-node-web-proxy \
        cgconfig \
        cgregd \
        openshift-cgroups \
        openshift-port-proxy \
```



```
openshift-gears
do
  chkconfig ${i} off
done

rm -rf /etc/mcollective/* /etc/openshift/*

}

# -----

host="roses-backup-node"

nics=( \
  "eth0 10.16.136.177 255.255.248.0" \
)

ip="10.16.136.177"
```

B.11 roses-create-app.sh

This script is executed on the broker host and is used after the roses-broker and roses-node systems are installed. It creates two applications. One called earth and one called mars.

```
#!/bin/bash

. roses-common.sh

# roses-create-app.sh

## Functions
#

rhc_setup() {

  output "Installing the Red Hat Cloud Client"
  yum -y install rhc

  rhc setup --clean --server roses-broker.example.org <<EOF
yes
${ose_user}
${ose_passwd}
yes
yes
${namespace}
EOF

}

rhc_create_php_app() {
  app_name=$1
```



```
output "Creating app: ${app_name}"

rhc create-app ${app_name} php-5.3
cd ${app_name}

cat <<EOF > php/index.php
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">
  <title>Welcome to Planet ${app_name}</title>
  <style>
    html {
      background: blue;
    }
    body {
      background: #333;
      color: white;
      font-family: "Helvetica Neue",Helvetica,"Liberation Sans",Arial,sans-serif;
      width: 40em;
      margin: 0 auto;
      padding: 3em;
    }
  </style>
</head>
<body>
  <h1>
    Welcome to Planet ${app_name}
  </h1>
</body>
</html>
EOF

git commit -a -m "Initial Page Push of Planet ${app_name}"

git push

cd ..
}

# -----

ose_user=rosesuser
ose_passwd=CHANGEME

namespace="planet"

# -----

(
```




```
# Check if a configuration file was passed on the command line.
# Source the file if there was.
#
if [ "$1" ]
then
  if [ -e $1 ]
  then
    . $1
  else
    output "Could not read input file $1."
    exit 1
  fi
fi

rhc_setup
rhc_create_php_app "earth"
rhc_create_php_app "mars"

) | tee roses-create-app.log
```

B.12 roses-node-prepare-shared.sh

This script is executed on the roses-node host and is used to move the gear and user data to the shared storage. Before this script is executed, the shared virtual disk must be attached to the roses-node virtual machine.

```
#!/bin/bash

. roses-common.sh

# roses-node-prepare-shared.sh

## Functions
#

share_prepare() {

  output "Creating preparing shared storage"

  parted --script ${shared_storage} "mklabel msdos"
  parted --script ${shared_storage} "mkpart primary 1024 5G"
  parted --script ${shared_storage} "mkpart primary 5G -1s"
  parted --script ${shared_storage} "print"

  mkfs.ext4 ${shared_storage}1
  mkfs.ext4 ${shared_storage}2

}

services_stop() {
  output "Stopping services"
  for i in mcollective \
    httpd \
```



```
ntpd \  
oddjobd \  
openshift-node-web-proxy \  
cgconfig \  
cgred \  
openshift-cgroups \  
openshift-port-proxy \  
openshift-gears  
do  
  service ${i} stop  
done  
  
}  
  
share_mv_dirs() {  
  
  output "Moving directories"  
  mkdir /var/lib/node_share  
  mount ${shared_storage}1 /var/lib/node_share  
  
  mkdir /var/lib/node_share/etc_openshift  
  mkdir /var/lib/node_share/etc_mcollective  
  
  shopt -s dotglob  
  mv /etc/openshift/* /var/lib/node_share/etc_openshift  
  mv /etc/mcollective/* /var/lib/node_share/etc_mcollective  
  
  chcon --reference /etc/openshift /var/lib/node_share/etc_openshift  
  chcon --reference /etc/mcollective /var/lib/node_share/etc_mcollective  
  
  mount ${shared_storage}2 /mnt  
  mv /var/lib/openshift/* /mnt  
  chcon --reference /var/lib/openshift /mnt  
  umount /mnt  
  
  echo "${shared_storage}1          /var/lib/node_share ext4 defaults"      >> /etc/fstab  
  echo "${shared_storage}2          /var/lib/openshift ext4 usrquota,defaults" >> /etc/fstab  
  echo "/var/lib/node_share/etc_openshift /etc/openshift  none bind,auto"      >> /etc/fstab  
  echo "/var/lib/node_share/etc_mcollective /etc/mcollective  none bind,auto"    >> /etc/fstab  
  
  mount /etc/openshift  
  mount /etc/mcollective  
  mount /var/lib/openshift  
}  
  
share_quotas() {  
  
  output "Moving quotas"  
  quotacheck --create-files --no-remount --user --group /var/lib/openshift  
  
  restorecon /var/lib/openshift/aquota.user
```



```

quotaon /var/lib/openshift

for user in $( ls /var/lib/openshift )
do
  if ls -dZ /var/lib/openshift/${user} | grep -q openshift_var_lib_t
  then
    t_quota=$( repquota / | grep ${user} | awk '{ print $4 " " $5 " " $7 " " $8}' )
    setquota --user ${user} ${t_quota} /var/lib/openshift
  fi
done

}

share_user_cron() {

  mkdir /etc/cron.5minutely
  chmod -v 755 /etc/cron.5minutely
  chcon --reference /etc/cron.hourly /etc/cron.5minutely

  cat <<'EOCS' > /etc/cron.5minutely/gear_user_info.sh
  #! /bin/bash

  if [ ! -d /var/lib/node_share/pam_limits ]
  then
    mkdir /var/lib/node_share/pam_limits
  fi

  > /var/lib/node_share/node_user_info.current

  for user in $( /bin/ls /var/lib/openshift )
  do
    if ls -dZ /var/lib/openshift/${user} | grep -q openshift_var_lib_t
    then
      cp -fv /etc/security/limits.d/*${user}.conf /var/lib/node_share/pam_limits

      t_uinfo=$( grep ${user} /etc/passwd | awk -F ':' '{print $3 " " $4}' )
      t_cgrps=$( grep ${user} /etc/cgrouprules.conf | awk '{print $2}' )

      echo "${user} ${t_uinfo} ${t_cgrps}" >> /var/lib/node_share/node_user_info.current
      echo "${user} ${t_uinfo} ${t_cgrps}" >> /var/lib/node_share/node_user_info.$( date +%Y%m )
    fi
  done
  EOCS

  chmod -v +x /etc/cron.5minutely/gear_user_info.sh

  cat <<'EOCT' > /etc/cron.d/5minutely
  SHELL=/bin/bash
  PATH=/sbin:/bin:/usr/sbin:/usr/bin
  MAILTO=root
  HOME=/

  */5 * * * * root run-parts /etc/cron.5minutely
  EOCT

```



```
chcon --reference /etc/cron.d/sysstat /etc/cron.d/5minutely
service crond reload

}

# -----

shared_storage="/dev/vdb"

# -----

(
# Check if a configuration file was passed on the command line.
# Source the file if there was.
#
if [ "$1" ]
then
    if [ -e $1 ]
    then
        . $1
    else
        output "Could not read input file $1."
        exit 1
    fi
fi

share_prepare
services_stop
share_mv_dirs
share_quotas
share_user_cron

init 6

) | tee roses-node-prepare-shared.log
```

B.13 roses-node-failover.sh

This script is used to complete the failover after the primary node host fails. This script is executed on the roses-backup-node after the shared storage disk is detached from the roses-node virtual machine and attached to the roses-node-failover virtual machine.

```
#!/bin/bash

. roses-common.sh

# roses-node-failover.sh
```



```
## Functions
```

```
#
```

```
services_on() {  
    output "Enabling services"  
    for i in mcollective \  
        httpd \  
        ntpd \  
        oddjobd \  
        openshift-node-web-proxy \  
        cgconfig \  
        cgred \  
        openshift-cgroups \  
        openshift-port-proxy \  
        openshift-gears  
    do  
        chkconfig ${i} on  
    done
```

```
}
```

```
share_present_mounts() {
```

```
    output "Moving directories"
```

```
    echo "${shared_storage}1          /var/lib/node_share ext4 defaults"      >> /etc/fstab  
    echo "${shared_storage}2          /var/lib/openshift ext4 usrquota,defaults" >> /etc/fstab  
    echo "/var/lib/node_share/etc_openshift /etc/openshift none bind,auto"      >> /etc/fstab  
    echo "/var/lib/node_share/etc_mcollective /etc/mcollective none bind,auto"   >> /etc/fstab
```

```
    mkdir /var/lib/node_share  
    mount /var/lib/node_share
```

```
    mount /etc/openshift  
    mount /etc/mcollective  
    mount /var/lib/openshift  
}
```

```
share_create_users() {
```

```
    output "Creating Users"
```

```
    while read id uid gid cg  
    do  
        groupadd --gid ${gid} ${id}  
        useradd --base-dir /var/lib/openshift -c "OpenShift guest" --gid ${gid} --uid ${uid} -M -s  
        /var/lib/openshift ${id}
```

```
        echo "${id}  ${cg}  /openshift/${id}" >> /etc/cgroles.conf
```

```
        cp /var/lib/node_share/pam_limits/${id}.conf /etc/security/limits.d
```



```
done < /var/lib/node_share/node_user_info.current

restorecon /etc/security/limits.d/*
}

share_user_cron() {

    mkdir /etc/cron.5minutely
    chmod -v 755 /etc/cron.5minutely
    chcon --reference /etc/cron.hourly /etc/cron.5minutely

    cat <<'EOCS' > /etc/cron.5minutely/gear_user_info.sh
    #! /bin/bash

    if [ ! -d /var/lib/node_share/pam_limits ]
    then
        mkdir /var/lib/node_share/pam_limits
    fi

    > /var/lib/node_share/node_user_info.current

    for user in $( /bin/ls /var/lib/openshift )
    do
        if ls -dZ /var/lib/openshift/${user} | grep -q openshift_var_lib_t
        then
            cp -fv /etc/security/limits.d/*${user}.conf /var/lib/node_share/pam_limits

            t_uinfo=$( grep ${user} /etc/passwd | awk -F ':' '{print $3 " " $4}' )
            t_cgrps=$( grep ${user} /etc/cgrouplists.conf | awk '{print $2}' )

            echo "${user} ${t_uinfo} ${t_cgrps}" >> /var/lib/node_share/node_user_info.current
            echo "${user} ${t_uinfo} ${t_cgrps}" >> /var/lib/node_share/node_user_info.$( date +%Y%m )
        fi
    done
    EOCS

    chmod -v +x /etc/cron.5minutely/gear_user_info.sh

    cat <<EOCT > /etc/cron.d/5minutely
    SHELL=/bin/bash
    PATH=/sbin:/bin:/usr/sbin:/usr/bin
    MAILTO=root
    HOME=/

    */5 * * * * root run-parts /etc/cron.5minutely
    EOCT

    chcon --reference /etc/cron.d/sysstat /etc/cron.d/5minutely
    service crond reload

}
```



```
# -----

host="roses-node"
domain="example.org"
nameservers="10.16.136.36"
gateway=10.16.143.254

nics=( \
    "eth0 10.16.136.74 255.255.248.0" \
)

shared_storage="/dev/vdb"

# -----

(
# Check if a configuration file was passed on the command line.
# Source the file if there was.
#
if [ "$1" ]
then
    if [ -e $1 ]
    then
        . $1
    else
        output "Could not read input file $1."
        exit 1
    fi
fi

yum -y update

conf_networking

share_present_mounts

share_create_users
share_user_cron

services_on

init 6

) | tee roses-node-failover.log
```



Appendix C: Contributors

Scott Collier, Senior Principal Software Engineer

Technical Assistance, Content, Review

Steve Reichard, Senior Principal Software Engineering

Technical Assistance

Balaji Jayavelu

Technical Assistance, Content

Luke Meyer, Senior Software Engineering

Technical Review



Appendix D: Revision History

Revision 1.0

Friday October 25, 2013

John Herr

Initial Release

