



Red Hat Reference Architecture Series

RHEV V3 Application Clustering inside RHEV Guests

Postfix and PostgreSQL

John Herr, Senior Software Engineer
RHCA, RHCVA

Version 1.0
April 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.

Dell is a registered trademark of Dell Inc.

IEEE is a registered trademark of The Institute of Electrical and Electronics Engineers, Incorporated.

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

© 2012 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 Red Hat Enterprise Virtualization.....	2
2.1 RHEV Hypervisor.....	2
2.2 Red Hat Enterprise Virtualization.....	3
2.3 High Availability Add-On.....	5
2.3.1 Quorum.....	5
2.3.2 Resource Group Manager.....	5
2.3.3 Fencing.....	6
2.3.4 CMAN.....	6
2.3.5 Ricci.....	7
2.3.6 CCS.....	7
3 Reference Architecture Environment.....	8
3.1 Overview.....	8
3.2 Infrastructure Systems Configuration.....	10
3.3 RHEV Configuration.....	12
3.4 Virtual Machine Configuration.....	13
4 Creating the Environment.....	19
4.1 Red Hat Enterprise Virtualization Infrastructure.....	19
4.2 Virtual Machine Creation.....	19
4.2.1 Hypervisor Pinning.....	20
4.2.2 Custom MAC Addresses.....	21
5 Virtual Machine Configuration.....	23
5.1 Red Hat Enterprise Virtualization Guest Agent.....	23
5.2 Network Time Service (NTP).....	24
5.3 Red Hat Enterprise Virtualization Guest Agent.....	25
5.4 Clustered Virtual Machines.....	26
5.4.1 Red Hat Enterprise Linux High Availability Add-On.....	26
5.4.2 Firewall Configuration.....	27
6 Clustering Virtual Machines Running PostgreSQL.....	28
6.1 Dynamic IP Address.....	28



6.2 Clustered Node Name Resolution.....	28
6.3 Primary Node.....	30
6.3.1 Shared Storage.....	30
6.3.2 Database Initialization.....	38
6.3.3 Firewall Configuration.....	40
6.3.4 SELinux.....	40
6.3.5 Cluster Configuration.....	42
6.3.5.1 Define Cluster and Add Nodes.....	43
6.3.5.2 Configure Fencing.....	43
6.3.5.3 Configure a Failover Domain.....	45
6.3.5.4 Define PostgreSQL Service.....	46
6.4 Secondary Node.....	48
6.4.1 Storage.....	48
6.4.2 Database Initialization.....	53
6.4.3 Firewall Configuration.....	53
6.4.4 SELinux.....	53
6.5 Starting the Cluster.....	54
7 Clustering Virtual Machines Running a rsyslog Server.....	57
7.1 Clustered Node Name Resolution.....	58
7.2 Dynamic IP Address.....	58
7.3 Primary Node.....	58
7.3.1 The rsyslog Daemon.....	58
7.3.1.1 Enable Logging to the Database.....	59
7.3.1.2 Allow Remote Logs to be Received.....	61
7.3.1.3 Restart the Service.....	62
7.3.1.4 Firewall Configuration.....	62
7.3.2 SELinux.....	63
7.3.3 Database Creation.....	64
7.3.4 Ensure rsyslog Logs to the Database.....	67
7.3.5 Cluster Configuration.....	68
7.3.5.1 Define a Cluster and Add Nodes.....	68
7.3.5.2 Configure Fencing.....	69
7.3.5.3 Configure a Failover Domain.....	69
7.3.5.4 Define rsyslog Service.....	69
7.4 Secondary Node.....	70
7.4.1 The rsyslog Daemon.....	70
7.4.1.1 Enable Logging to the Database.....	70



7.4.1.2 Allow Remote Logs to be Received.....	70
7.4.1.3 Restart the Service.....	70
7.4.1.4 Firewall Configuration.....	71
7.4.2 SELinux.....	71
7.5 Starting the Cluster.....	72
7.6 Clients.....	73
7.6.1 UDP Client.....	73
7.6.2 TCP Client.....	74
7.6.3 RELP Client.....	76
8 Testing Fail Over.....	78
9 Conclusion.....	81
Appendix A: Configuration Files.....	82
Appendix B: Channel Subscriptions.....	83
Appendix C: Resources.....	84
Appendix D: Contributors.....	85
Appendix E: Revision History.....	86



1 Executive Summary

Ensuring the availability of production services is critical in today's IT environment. A service that is unavailable for even an hour can be extremely costly to a business. Add this requirement with the push to reduce the costs of running physical systems, it becomes difficult to implement the needs of the business.

A Red Hat Enterprise Virtualization environment implementing virtual machines running the Red Hat Enterprise Linux High Availability Add-On provides a solution to this issue. Virtualizing physical servers and then clustering them across multiple hypervisors, provides a resource sharing and highly available solution.

This reference architecture stands up a central logging server using the `rsyslog` logger to write log entries to a PostgreSQL database. Both the logging server and database run in separate two-node active/passive clustered virtual machines. The virtual machines are configured to run on dedicated hypervisors, thus ensuring neither of the clustered nodes run on the same hypervisor.

The goal of this reference architecture is to provide the reader with an understanding of using the Red Hat Enterprise Linux High Availability Add-On running on virtual machines withing a Red Hat Enterprise Virtualization environment.



2 Red Hat Enterprise Virtualization

2.1 RHEV 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.

Running fully virtualized guests, i.e., guests with unmodified guest operating systems, used to require complex hypervisors and previously incurred a performance penalty for emulation and translation of I/O and memory requests.

Over the last few years chip vendors Intel and AMD have been steadily adding CPU features that offer hardware enhancements to support virtualization. Most notable are:

1. First-generation hardware assisted virtualization: Removes the requirement for hypervisor to scan and rewrite privileged kernel instructions using Intel VT (Virtualization Technology) and AMD's SVM (Secure Virtual Machine) technology.
2. Second-generation hardware assisted virtualization: Offloads virtual to physical memory address translation to CPU/chip-set using Intel EPT (Extended Page Tables) and AMD RVI (Rapid Virtualization Indexing) technology. This provides significant reduction in memory address translation overhead in virtualized environments.
3. Third-generation hardware assisted virtualization: Allows PCI I/O devices to be attached directly to virtual machines using Intel VT-d (Virtualization Technology for directed I/O) and AMD IOMMU. Also, SR-IOV (Single Root I/O Virtualization) which allows special PCI devices to be split into multiple virtual devices. This provides significant improvement in guest I/O performance.

The great interest in virtualization has led to the creation of several different hypervisors. However, many of these pre-date hardware-assisted virtualization, and are therefore somewhat complex pieces of software. With the advent of the above hardware extensions, writing a hypervisor has become significantly easier and it is now possible to enjoy the benefits of virtualization while leveraging existing open source achievements to date.

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.

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



2.2 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.



Figure 2.2.1: RHEV Environment provides a graphical representation of a typical Red Hat Enterprise Virtualization environment with each component listed.

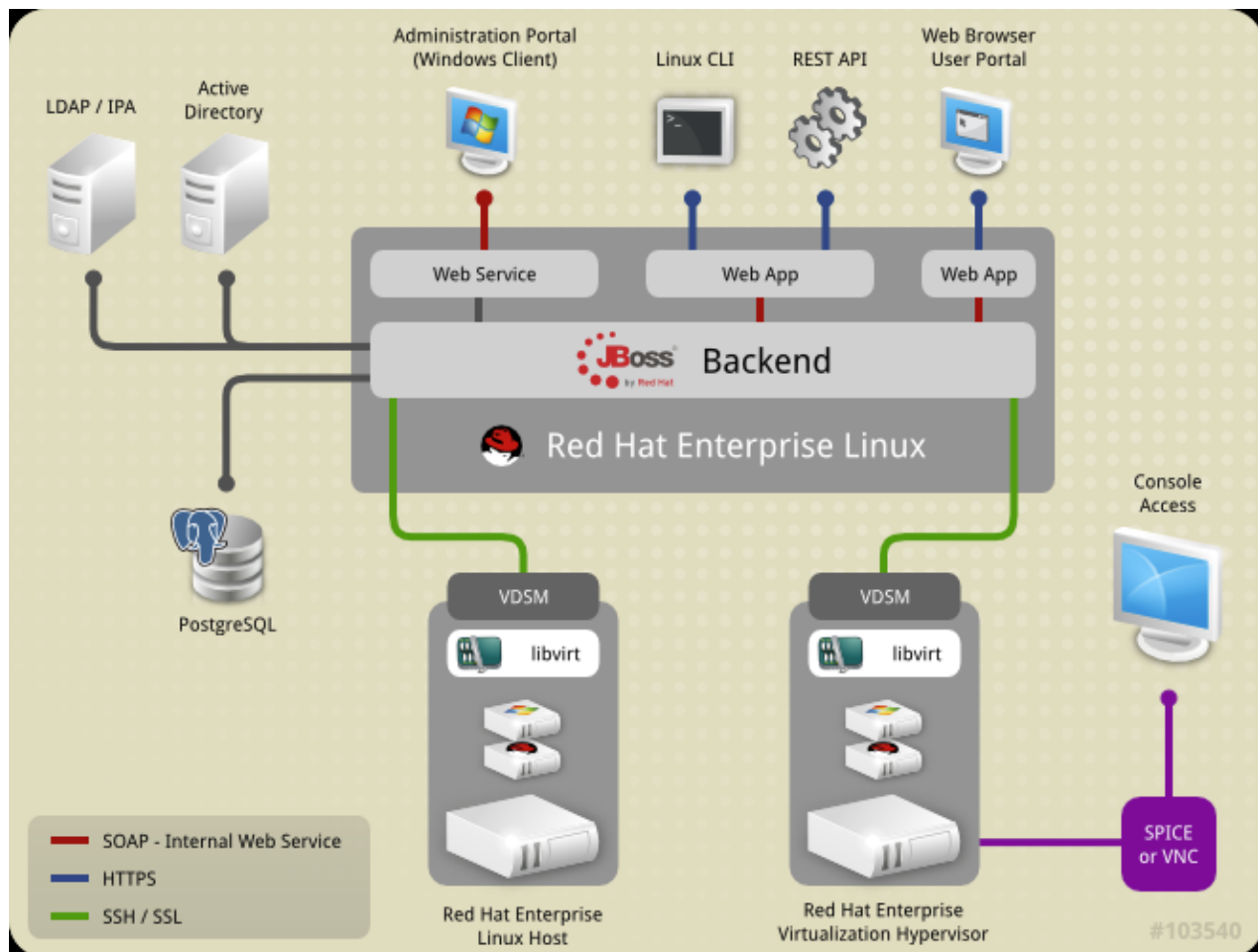


Figure 2.2.1: RHEV Environment



2.3 High Availability Add-On

The *High Availability Add-On* for Red Hat Enterprise Linux provides high availability of services by eliminating single points of failure. By offering fail over services between nodes within a cluster, the *High Availability Add-On* supports high availability for up to 16 nodes. (Currently this capability is limited to a single LAN or datacenter located within one physical site.)

The *High Availability Add-On* also enables fail over for off-the-shelf applications such as Apache, MySQL, PostgreSQL and Samba, any of which can be coupled with resources like IP addresses and single-node file systems to form highly available services. The *High Availability Add-On* can also be easily extended to any user-specified application that is controlled by an init script per UNIX System V (SysV) standards.

When using the *High Availability Add-On*, a highly available service can fail over from one node to another with no apparent interruption to cluster clients. The *High Availability Add-On* ensures data integrity when one cluster node takes over control of a service from another cluster node. It achieves this by promptly evicting nodes from the cluster that are deemed to be faulty using a method called "fencing", thus preventing data corruption. The *High Availability Add-On* supports several types of fencing, including both power and storage area network (SAN) based fencing.

The following sections describe the various components of the *High Availability Add-On* in the context of this reference architecture and the deployment of clustered Samba.

2.3.1 Quorum

Quorum is a voting algorithm used by the cluster manager (CMAN). To maintain *quorum*, the nodes in the cluster must agree about their status among themselves. *Quorum* determines which nodes in the cluster are dominant. For example, if there are three nodes in a cluster and one node loses connectivity, the other two nodes communicate with each other and determine that the third node needs to be fenced. The action of fencing ensures that the node which lost connectivity does not corrupt data.

By default each node in the cluster has one *quorum* vote, although this is configurable. There are two methods the nodes can communicate with each other to determine *quorum*. The first method *quorum* via network consists of a simple majority (50% of the nodes +1 extra). The second method is by adding a *quorum* disk. The *quorum* disk allows for user-specified conditions to exist which help determine which node(s) should be dominant.

This reference architecture uses network *quorum* - a dedicated *quorum* disk is not required.

2.3.2 Resource Group Manager

Resource group manager (*rgmanager*) provides fail over capabilities for collections of cluster resources known as resource groups or resource trees. *Rgmanager* allows system administrators to define, configure, and monitor cluster services such as **httpd** or **mysql**. In the event of a node failure, *rgmanager* relocates the clustered service to another node to restore service availability. Services can also be restricted to run on specific cluster nodes.



2.3.3 Fencing

Fencing is the disconnection of a node from the cluster's shared storage. Fencing prevents the affected node from issuing I/O to shared storage, thus ensuring data integrity. The cluster infrastructure performs fencing through *fenced*, the fence daemon.

When CMAN determines that a node has failed, it communicates to other cluster-infrastructure components to inform them that the node has failed. The failed node is fenced when *fenced* is notified. Other cluster-infrastructure components determine what actions to take - that is, they perform any recovery that needs to be done. For example, distributed lock manager (*DLM*) and Global File System version 2 (*GFS2*), when notified of a node failure, suspend activity until they detect that *fenced* has completed fencing the failed node. Upon confirmation that the failed node is fenced, *DLM* and *GFS2* perform recovery. *DLM* releases locks of the failed node; *GFS2* recovers the journal of the failed node.

The fencing program (*fenced*) determines from the cluster configuration file which fencing method to use. Two key elements in the cluster configuration file define a fencing method: fencing agent and fencing device. The fencing program makes a call to a fencing agent specified in the cluster configuration file. The fencing agent, in turn, fences the node via a fencing device. When fencing is complete, the fencing program notifies the cluster manager. The High Availability Add-On provides a variety of fencing methods:

- Power fencing - A fencing method that uses a power controller to power off an inoperable node
- Storage fencing - Includes fencing methods that disable the Fibre Channel port that connects storage to an inoperable node. SCSI-3 persistent reservations are another commonly used storage fencing method in which access to a common shared storage device can be revoked to an inoperable node.
- Systems management fencing - Fencing methods that disable I/O or power to an inoperable node. Examples include IBM® BladeCenter, Dell® DRAC/MC, HP® ILO, IPMI, and IBM RSA II.

2.3.4 CMAN

CMAN manages cluster membership, fencing, locking and quorum. CMAN runs as a service on all cluster nodes and simplifies the management of the following HA cluster daemons:

- corosync (manages cluster membership, messaging, quorum)
- fenced (manages cluster node I/O fencing)
- dlm_controld (manages distributed file locking to shared file systems)
- gfs_controld (manages GFS2 file system mounting and recovery)

From a systems management perspective, CMAN is the first service in the component stack started when bringing up a clustered Samba node.



2.3.5 Ricci

Ricci is the cluster management and configuration daemon that runs on the cluster nodes. When **ricci** is installed it creates a user account called **ricci** and a password is set for the account. All **ricci** accounts must be configured with the same password across all cluster nodes to allow authentication with the **luci** management server. The **ricci** daemon requires port 11111 to be open for both tcp and udp traffic.

2.3.6 CCS

The Cluster Configuration System (CCS) was introduced in Red Hat Enterprise Linux 6.1. CCS provides a powerful way of managing a Red Hat Enterprise Linux cluster from the command line. CCS allows an administrator to create, modify and view cluster configurations from a remote node through **ricci** or on a local file system. Table 2.2.6: Common CCS Switches

Cluster configuration details are stored as XML entries in the file `/etc/cluster/cluster.conf`. To avoid configuration errors, changes to this file should always be done through the `ccs` utility.

CCS authenticates with **ricci** agents by using automatically generated certificate files in `~/.ccs/`. These files allow CCS to communicate with **ricci** securely over Secure Socket Layer (SSL) encrypted links. To communicate with **ricci** the password for the **ricci** agent on each cluster node must be known by the administrator as CCS prompts for the password at the first connection.



3 Reference Architecture Environment

3.1 Overview

The environment for this reference architecture as depicted in **Figure 3.1.1: Environment**, consists of a Red Hat Enterprise Virtualization Manager connected to the public or rhevm network along with an NFS server. These two systems are virtual machines running on a Red Hat Enterprise Linux 6 KVM server. The NFS server exports the Import and Export domains for the Red Hat Enterprise Virtualization environment.

The Red Hat Enterprise Virtualization Manager manages two Red Hat Enterprise Virtualization hypervisors. These hypervisors have three network adapters connected to different networks. One adapter is connected to the public or rhevm network, another adapter is connected to a storage network and is used to access a Dell MD3000i iSCSI target. The final network is used to pass cluster management traffic.

Virtual machines are clustered together using the Red Hat Enterprise Linux High Availability Add-On packages. Two separate clusters are created, each with two virtual machines. Each cluster runs in an active-passive configuration and contains a primary cluster node and a secondary cluster node. The clustered services are configured to run on the primary cluster node when it is available. Each primary node is pinned to a different hypervisor and each secondary node is pinned to the opposite hypervisor. This prevents a single hypervisor from running all the cluster members. This is done to achieve better performance and high availability.

One cluster creates a highly available instance of a PostgreSQL database, while the second cluster is used to create a logging server instance that writes log events to the PostgreSQL database.



Three virtual machines are created to act as logging clients. Each uses a different protocol to log events to the logging server. The three protocols used are UDP, TCP, and RELP.

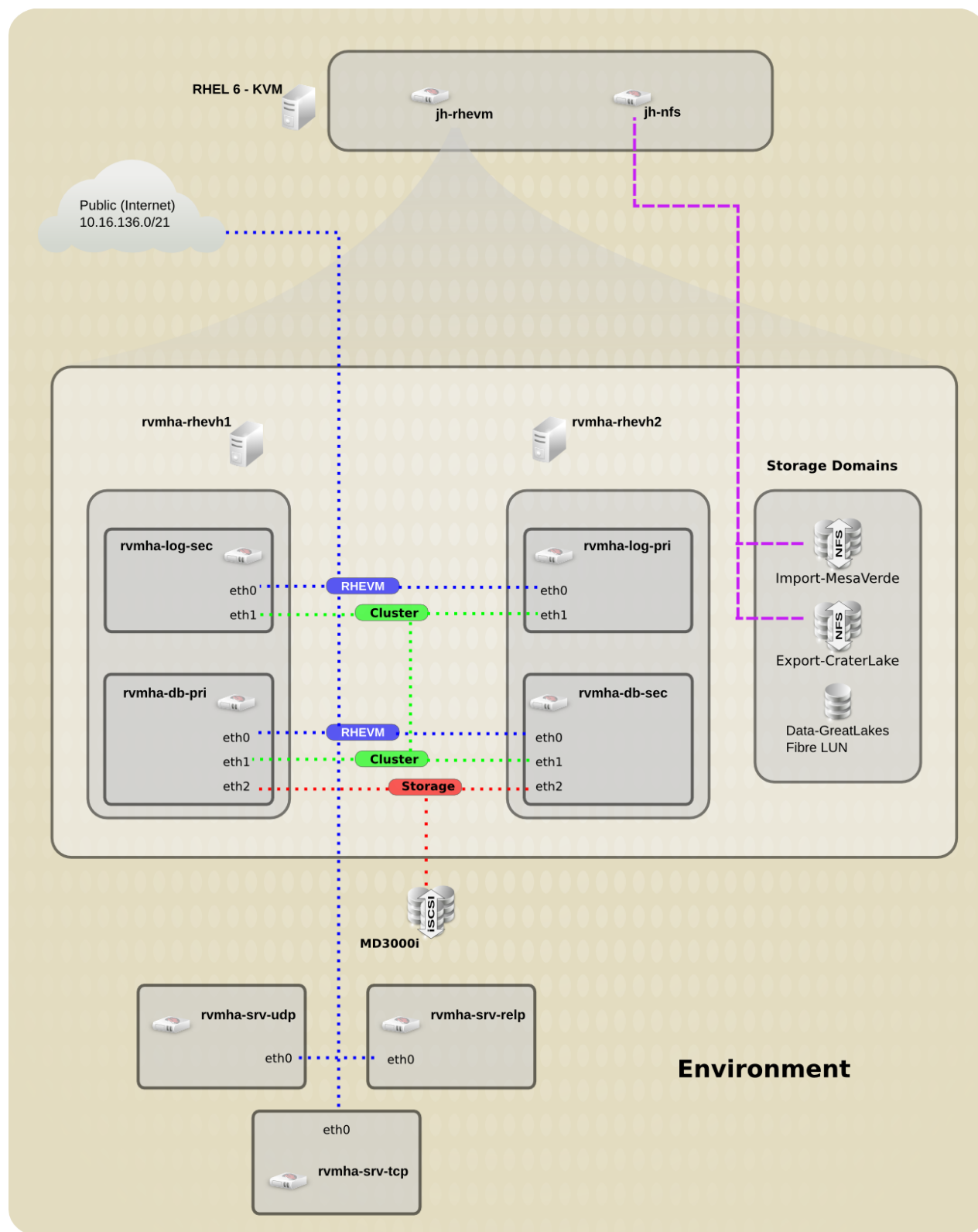


Figure 3.1.1: Environment



3.2 Infrastructure Systems Configuration

Table 3.2.1: Server Configurations and **Table 3.2.2: Storage Devices** contain the configuration information for the infrastructure systems used in the writing of this reference architecture.

Server	Configuration
jh-rhevms [KVM Virtual Machine]	RHEL 6.4 kernel 2.6.32-358.2.1.el6.x86_64
	SELinux Targeted Enforcing selinux-policy-3.7.19-195.el6_4.3.noarch
	rhevms-3.1.0-50.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
jh-nfs [KVM Virtual Machine]	RHEL 6.4 kernel 2.6.32-358.2.1.el6.x86_64
	SELinux Targeted Enforcing selinux-policy-3.7.19-195.el6_4.3.noarch
	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



Server	Configuration
ravmha-rhevh1 [HP Proliant BL460c G6]	RHEV Hypervisor 6.4-20130318.1.el6_4
	2 x Quad Core Intel Xeon CPU X550 @ 2.67GHz
	48 GB Memory
	2 x 146 GB SAS Internal Disk Drives (Mirrored)
	2 x QLogic ISP2532-based 8 Gb FC HBA 1 x 500 GB LUN presented by an HP StorageWorks MSA2324fc array (Shared with ravmha-rhevh2)
	3 x Broadcom NetExtreme II BCM57711E Flex-10 10GB Ethernet Controller
ravmha-rhevh2 [HP Proliant BL460c G6]	RHEV Hypervisor 6.4-20130318.1.el6_4
	2 x Quad Core Intel Xeon CPU X550 @ 2.67GHz
	48 GB Memory
	2 x 146 GB SAS Internal Disk Drives (Mirrored)
	2 x QLogic ISP2532-based 8 Gb FC HBA 1 x 500 GB LUN presented by an HP StorageWorks MSA2324fc array (Shared with ravmha-rhevh1)
	3 x Broadcom NetExtreme II BCM57711E Flex-10 10GB Ethernet Controller

Table 3.2.1: Server Configurations

Storage Device	Configuration	
Dell MD3600i 10G iSCSI	IP Address	172.31.143.202
	Port	3260
	Firmware	07.80.41.60
	NVSRAM	N26X0-780890-901
	AMW	10.80.G6.47

Table 3.2.2: Storage Devices



3.3 RHEV Configuration

The Red Hat Enterprise Virtualization environment consisted of a single cluster titled NorthAmerica that is in the Earth data center. Three networks called rhevm, cluster, and storage are configured in the cluster. See **Table 3.3.2: Networks** for a brief explanation of each network.

The data center has three storage domains configured: a data domain, an export domain, and an ISO domain. **Table 3.3.3: Storage Domains** displays the configuration of the storage domains. Although configured, the export domain and the ISO domain are not used for this reference architecture.

Each hypervisor is configured for power management. Configuring power management on the hypervisors allows the Red Hat Enterprise Virtualization environment to power cycle the hypervisors if needed. See **Table 3.3.4: Power Management** for the power management configuration.

Item	Name
Data Center	Earth
Cluster	NorthAmerica

Table 3.3.1: Data Center Configuration

Network	Description
rhevm	Public network, used for all non-cluster and and non-storage traffic.
cluster	User for cluster nodes to communicate cluster traffic.
storage	ISCSI storage traffic.

Table 3.3.2: Networks

Name	Target
Data-GreatLakes	Fibre LUN
Export-CraterLake	jh-nfs.cloud.lab.eng.bos.redhat.com:/exports/EXPORT_CraterLake
ISO-MesaVerde	jh-nfs.cloud.lab.eng.bos.redhat.com:/exports/ISO_MesaVerde

Table 3.3.3: Storage Domains



Hypervisor	IP	User	Password	Type	Secure
rvmha-rhevh1	10.16.143.237	[USER]	[PASSWORD]	ILO	Checked
rvmha-rhevh2	10.16.143.238	[USER]	[PASSWORD]	ILO	Checked

Table 3.3.4: Power Management

3.4 Virtual Machine Configuration

Table 3.4.1: Database Servers Configuration, **Table 3.4.2: rsyslog Servers Configuration**, and **Table 3.4.3: Logging Clients Configuration** contain the configuration information for the virtual machines created in this reference architecture.

Virtual Machine	Configuration			
rvmha-db-pri	Red Hat Enterprise Linux Server release 6.4 (Santiago) 2.6.32-358.2.1.el6.x86_64			
	SELinux Targeted Enforcing selinux-policy-3.7.19-195.el6_4.3.noarch			
	cman-3.0.12.1-49.el6 ricci-0.16.2-63.el6 postgresql-server-8.4.13-1.el6_3 postgresql-8.4.13-1.el6_3 policycoreutils-2.0.83-19.30.el6			
	Hypervisor: rvmha-rhevh1			
	1 x Virtual CPU			
	1024 MB Memory			
	1 x VirtIO Disk Device @ 16GB			
	3 x VirtIO Network Adapters			
	Virtual Nic Name	Network	Hardware Address	IP Address
	public	rhevm	02:00:00:10:10:10	10.16.136.65
	mgmt	cluster	02:00:00:10:10:20	192.168.0.65
	iscsi	storage	02:00:00:10:10:30	172.31.0.65



Virtual Machine	Configuration			
rvmha-db-sec	Red Hat Enterprise Linux Server release 6.4 (Santiago) 2.6.32-358.2.1.el6.x86_64			
	SELinux Targeted Enforcing selinux-policy-3.7.19-195.el6_4.3.noarch			
	cman-3.0.12.1-49.el6 ricci-0.16.2-63.el6 postgresql-server-8.4.13-1.el6_3 postgresql-8.4.13-1.el6_3 policycoreutils-2.0.83-19.30.el6			
	Hypervisor: rvmha-rhevh2			
	1 x Virtual CPU			
	1024 MB Memory			
	1 x VirtIO Disk Device @ 16GB			
	3 x VirtIO Network Adapters			
	Virtual Nic Name	Network	Hardware Address	IP Address
	public	rhevm	02:00:00:10:20:10	10.16.136.66
	mgmt	cluster	02:00:00:10:20:20	192.168.0.66
	iscsi	storage	02:00:00:10:20:30	172.31.0.66

Table 3.4.1: Database Servers Configuration



Virtual Machine	Configuration			
rvmha-log-pri	Red Hat Enterprise Linux Server release 6.4 (Santiago) 2.6.32-358.2.1.el6.x86_64			
	SELinux Targeted Enforcing selinux-policy-3.7.19-195.el6_4.3.noarch			
	cman-3.0.12.1-49.el6 ricci-0.16.2-63.el6 rsyslog-5.8.10-6.el6 rsyslog-pgsql-5.8.10-6.el6 rsyslog-relp-5.8.10-6.el6 policycoreutils-2.0.83-19.30.el6			
	Hypervisor: rvmha-rhevh2			
	1 x Virtual CPU			
	1024 MB Memory			
	1 x VirtIO Disk Device @ 16GB			
	2 x VirtIO Network Adapters			
	Virtual Nic Name	Network	Hardware Address	IP Address
	public	rhevm	02:00:00:30:10:10	10.16.136.67
	mgmt	cluster	02:00:00:30:10:20	192.168.0.67



Virtual Machine	Configuration			
rvmha-log-sec	Red Hat Enterprise Linux Server release 6.4 (Santiago) 2.6.32-358.2.1.el6.x86_64			
	SELinux Targeted Enforcing selinux-policy-3.7.19-195.el6_4.3.noarch			
	cman-3.0.12.1-49.el6 ricci-0.16.2-63.el6 rsyslog-5.8.10-6.el6 rsyslog-pgsql-5.8.10-6.el6 rsyslog-relp-5.8.10-6.el6 policycoreutils-2.0.83-19.30.el6			
	Hypervisor: rvmha-rhevh1			
	1 x Virtual CPU			
	1024 MB Memory			
	1 x VirtIO Disk Device @ 16GB			
	2 x VirtIO Network Adapters			
	Virtual Nic Name	Network	Hardware Address	IP Address
	public	rhevm	02:00:00:30:20:10	10.16.136.68
	mgmt	cluster	02:00:00:30:20:20	192.168.0.68

Table 3.4.2: rsyslog Servers Configuration



Virtual Machine	Configuration			
rvmha-srv-udp	Red Hat Enterprise Linux Server release 6.4 (Santiago) 2.6.32-358.2.1.el6.x86_64			
	SELinux Targeted Enforcing selinux-policy-3.7.19-195.el6_4.3.noarch			
	rsyslog-5.8.10-6.el6rsyslog-5.8.10-6.el6 selinux-policy-3.7.19-195.el6_4.3 selinux-policy-targeted-3.7.19-195.el6_4.3			
	Hypervisor: Any			
	1 x Virtual CPU			
	1024 MB Memory			
	1 x VirtIO Disk Device @ 16 GB			
	1 x VirtIO Network Adapters			
	Virtual Nic Name	Network	Hardware Address	IP Address
	public	rhevm	02:00:00:40:10:30	10.16.136.73
rvmha-srv-tcp	Red Hat Enterprise Linux Server release 6.4 (Santiago) 2.6.32-358.2.1.el6.x86_64			
	SELinux Targeted Enforcing selinux-policy-3.7.19-195.el6_4.3.noarch			
	rsyslog-5.8.10-6.el6 selinux-policy-3.7.19-195.el6_4.3 selinux-policy-targeted-3.7.19-195.el6_4.3			
	Hypervisor: Any			
	1 x Virtual CPU			
	1024 MB Memory			
	1 x VirtIO Disk Device @ 16 GB			
	1 x VirtIO Network Adapters			
	Virtual Nic Name	Network	Hardware Address	IP Address
	public	rhevm	02:00:00:40:10:20	10.16.136.72



Virtual Machine	Configuration			
rvmha-srv-relp	Red Hat Enterprise Linux Server release 6.4 (Santiago) 2.6.32-358.2.1.el6.x86_64			
	SELinux Targeted Enforcing selinux-policy-3.7.19-195.el6_4.3.noarch			
	rsyslog-5.8.10-6.el6 rsyslog-relp-5.8.10-6.el6 selinux-policy-3.7.19-195.el6_4.3 selinux-policy-targeted-3.7.19-195.el6_4.3			
	1 x Virtual CPU			
	1024 MB Memory			
	1 x VirtIO Disk Device @ 16 GB			
	1 x VirtIO Network Adapters			
	Virtual Nic Name	Network	Hardware Address	IP Address
	public	rhevm	02:00:00:40:10:10	10.16.136.71

Table 3.4.3: Logging Clients Configuration



4 Creating the Environment

When creating the infrastructure environment and the virtual machines, refer to the Red Hat Enterprise Virtualization 3 documentation², specifically the Hypervisor Deployment Guide and Administration Guide for information on how to install and create the environment if needed.

4.1 Red Hat Enterprise Virtualization Infrastructure

The installation and configuration of the virtualization infrastructure is a standard installation without any special configurations done. Install and configure the Red Hat Enterprise Virtualization environment using the information in **Table 3.2.1: Server Configurations** and **Table 3.4.1: Database Servers Configuration**.

4.2 Virtual Machine Creation

After the infrastructure environment is installed and configured, the virtual machines are installed using the information supplied in **Table 3.4.1: Database Servers Configuration**, **Table 3.4.2: rsyslog Servers Configuration**, and **Table 3.4.3: Logging Clients Configuration**.

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



4.2.1 Hypervisor Pinning

During the creation of the virtual machines, make sure a specific host is selected for the virtual machine to run on and that the virtual machine only runs on the selected host. If this is not done, then the High Availability Add-On clustering software can start the virtual machine on an incorrect hypervisor when fencing occurs. The tables containing the virtual machine configuration indicate which hypervisor the virtual machine is pinned to. See **Figure 4.2.1.1: Hypervisor Pinning** for a sample screen of the host configuration tab.

Edit Server Virtual Machine

General
Initial Run
Console
Host
High Availability
Resource Allocation
Boot Options
Custom Properties

Run On:

☐ Any Host in Cluster

☒ Specific rvmha-rhevh1.cloud.lab.eng.bos.redh

Run/Migration Options:

☒ Run VM on the selected host (no migration allowed)

☐ Allow VM migration only upon Administrator specific request (system will not trigger automatic migration of this VM)

CPU Pinning topology

Format: v#p[_v#p]
Examples:
• 0#0 => pin vCPU 0 to pCPU 0
• 0#0_1#3 => pin vCPU 0 to pCPU 0 and pin vCPU 1 to pCPU 3
• 1#1-4,^2 => pin vCPU 1 to pCPU set 1 to 4, excluding 2

OK Cancel

Figure 4.2.1.1: Hypervisor Pinning



4.2.2 Custom MAC Addresses

The virtual servers used in this environment are installed over the network using **Cobbler** and the **Red Hat Satellite Server**. To perform the installations, the systems are powered on and told to perform a PXE installation.

During installation, the configuration of the operating system is determined by the MAC (hardware) address of the system. Because of this, the hardware addresses of the network interfaces must be known prior to installing the virtual machines. A custom hardware address is assigned to each interface during the creation of the virtual network interfaces.

Hardware address can be *UNIVERSALLY* or *LOCALLY* administered. Universally managed hardware addresses are assigned to each adapter that connects to a network. These hardware addresses can be used on public networks and are the addresses most people are familiar with. Locally administered addresses can be assigned and used within a closed network environment. These addresses should not be used on public networks. As with all hardware addresses, each network adapter must have a unique hardware addresses relative to the scope of the network it is connected to.

If the second LSB (Least Significant Bit) of the first octet of the hardware address is 0, then the address is universally administered. If this bit is one, then the address is locally administered. Refer to the IEEE 802[®] standard³ for more information.

This reference architecture uses locally administered hardware addresses for the virtual machine interfaces. To keep the address calculations simple, zeros are placed in all bits of the first octet except the second LSB. This means the first octet of all hardware addresses for the virtual network interfaces is 02.

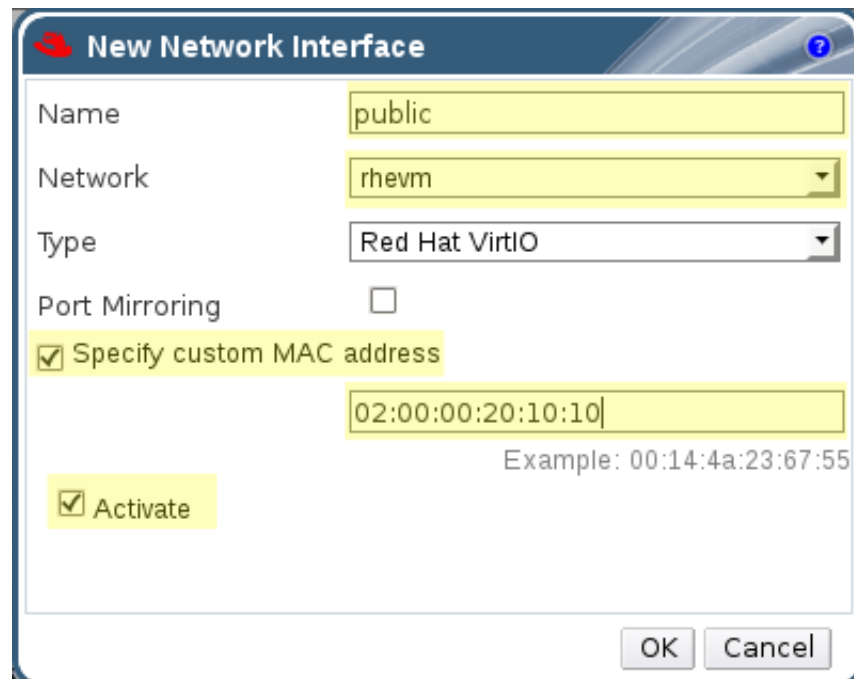
Use the information in **Table 3.4.1: Database Servers Configuration**, **Table 3.4.2: rsyslog Servers Configuration**, and **Table 3.4.3: Logging Clients Configuration** to assign a known hardware address to each virtual network interface of the created virtual machines.

Using **Figure 4.2.2.1: Custom Hardware Address** as reference, place a check mark in the box next to **Specify custom MAC address**. Enter the appropriate hardware address for the virtual machine in the box to the right and below of the check box.

³ <http://standards.ieee.org/getieee802/download/802-2001.pdf>



Place a check mark in the box next to **Activate** and select **OK**. The virtual NIC is now created and assigned to the virtual machine.



The image shows a 'New Network Interface' dialog box. It has a title bar with a red hat icon and a question mark. The fields are: Name (public), Network (rhevm), Type (Red Hat VirtIO), Port Mirroring (unchecked), Specify custom MAC address (checked), and a MAC address field (02:00:00:20:10:10). There is an 'Activate' checkbox checked. An example MAC address '00:14:4a:23:67:55' is shown. At the bottom are 'OK' and 'Cancel' buttons.

Name	public
Network	rhevm
Type	Red Hat VirtIO
Port Mirroring	<input type="checkbox"/>
<input checked="" type="checkbox"/> Specify custom MAC address	
	02:00:00:20:10:10
	Example: 00:14:4a:23:67:55
<input checked="" type="checkbox"/> Activate	

OK Cancel

Figure 4.2.2.1: Custom Hardware Address



5 Virtual Machine Configuration

The virtual servers used in this environment are installed over the network using **Cobbler** and the **Red Hat Satellite Server**. To perform the installations, the systems are powered on and told to perform a PXE installation.

The details of setting up this installation environment is beyond the scope of this reference architecture. Be aware that any supported method could be used to install the systems.

After installation, the networking is configured using the information in **Table 3.4.1: Database Servers Configuration**, **Table 3.4.2: rsyslog Servers Configuration**, and **Table 3.4.3: Logging Clients Configuration**.

After the virtual machines are installed and networking is configured, perform the steps in this section on all virtual machines used in the environment.

5.1 Red Hat Enterprise Virtualization Guest Agent

The virtual machines must subscribe to the `RHEL-X86_64-RHEV-AGENT-6-SERVER` channel to have access to the Red Hat Enterprise Virtualization Guest Agent. See **Appendix B: Channel Subscriptions** for the repository names if using Red Hat Subscription Manager.

Subscribe the virtual machines to the `RHEL-X86_64-RHEV-AGENT-6-SERVER` channel by issuing the **`rhn-channel`** command on each virtual machine.

The **`rhn-channel`** command uses two options to add channels. The **`--add`** option informs the command that it is to subscribe the system to the channel specified by the **`--channel`** option. The **`--channel`** option takes a single channel name as an argument and must be specified for each channel needing subscribed to.

```
# rhn-channel --add --channel rhel-x86_64-rhev-agent-6-server
Username: [User]
Password: [Password]

# rhn-channel -l
rhel-x86_64-rhev-agent-6-server
rhel-x86_64-server-6
```



5.2 Network Time Service (NTP)

All the servers installed in this environment are configured to run the NTP service to synchronize the clocks. The following steps are performed on each of the virtual machines.

Install the **ntp** package.

```
# yum -y install ntp
Loaded plugins: product-id, rhnplugin, security, subscription-manager
This system is not registered to Red Hat Subscription Management. You can
use subscription-manager to register.
This system is receiving updates from RHN Classic or RHN Satellite.
Setting up Install Process

[ ... Output Abbreviated ... ]

Installing:
  ntp          x86_64          4.2.4p8-3.el6          rhel-x86_64-server-6          444 k

[ ... Output Abbreviated ... ]

Installed:
  ntp.x86_64 0:4.2.4p8-3.el6

Complete!
```

Edit the `/etc/ntp.conf` file and replace the server lines with the names or IP addresses of the time servers to be used. This environment uses the following:

```
server ns1.bos.redhat.com
server ns2.bos.redhat.com
```

Perform an initial sync of the time using the **ntpdate** command.

```
# ntpdate ns1.bos.redhat.com
1 Apr 10:07:46 ntpdate[1684]: adjust time server 10.16.255.2 offset
0.344996 sec
```

start the **ntp** daemon and ensure it starts at reboot.

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

# chkconfig ntpd on
```



5.3 Red Hat Enterprise Virtualization Guest Agent

The **rhev-guest-agent** package allows the hypervisor to better interact with the virtual machine by allowing it to issue commands to the underlying operating system.

Install the agent on all the virtual machines.

```
# yum -y install rhevm-guest-agent
Loaded plugins: product-id, rhnplugin, security, subscription-manager
This system is not registered to Red Hat Subscription Management. You can
use subscription-manager to register.
This system is receiving updates from RHN Classic or RHN Satellite.
Setting up Install Process
```

[... Output Abbreviated ...]

```
Installing:
rhev-guest-agent x86_64 1.0.5-8.el6ev    rhel-x86_64-rhev-agent-6-server
46 k
```

[... Output Abbreviated ...]

```
Installed:
rhev-guest-agent.x86_64 0:1.0.5-8.el6ev
```

Complete!

Start the agent and ensure it starts at boot.

```
# service ovirt-guest-agent start
Starting ovirt-guest-agent: [ OK ]

# chkconfig ovirt-guest-agent on
```



5.4 Clustered Virtual Machines

This section configures the Red Hat Enterprise Linux High Availability Add-On. All servers used as a cluster node must subscribe to the `RHEL-X86_64-SERVER-HA-6` channel to have access to the high availability clustering software. For the software to function correctly, configuration of the firewall must also be performed.

5.4.1 Red Hat Enterprise Linux High Availability Add-On

The *High Availability* group within **yum** provides the services and commands required to create, run and administrate a set of clustered servers. All the virtual machines must install the packages available from **yum** in the *High Availability* group.

Subscribe the virtual machines to the `RHEL-X86_64-SERVER-HA-6` channel by issuing the **rhn-channel** command on each virtual machine.

```
# rhn-channel --add --channel rhel-x86_64-server-ha-6
Username: [User]
Password: [Password]

# rhn-channel -l
rhel-x86_64-rhev-agent-6-server
rhel-x86_64-server-6
rhel-x86_64-server-ha-6
```

Install the packages from the *High Availability* group.

```
# yum -y groupinstall "High Availability"
Loaded plugins: product-id, rhnplugin, security, subscription-manager
This system is not registered to Red Hat Subscription Management. You can
use subscription-manager to register.
This system is receiving updates from RHN Classic or RHN Satellite.
rhel-x86_64-server-ha-6 | 1.5 kB 00:00
rhel-x86_64-server-ha-6/primary | 81 kB 00:00
rhel-x86_64-server-ha-6
253/253
Setting up Group Process
rhel-x86_64-server-ha-6/group | 18 kB 00:00
Resolving Dependencies

[ ... Output Abbreviated ... ]

Installed:
  ccs.x86_64 0:0.16.2-63.el6          cman.x86_64 0:3.0.12.1-49.el6
  omping.x86_64 0:0.0.4-1.el6        rgmanager.x86_64 0:3.0.12.1-17.el6

[ ... Output Abbreviated ... ]

Complete!
```




The **ricci** package is installed with the *High Availability* package group. This package provides a daemon of the same name. This daemon allows the cluster configuration commands to communicate with each node of the cluster.

The daemon runs as the user *RICCI* and a password must be set for the user.

Set a password for the ricci user.

```
# passwd ricci
Changing password for user ricci.
New password: [Password]
Retype new password: [Password]
passwd: all authentication tokens updated successfully.
```

Start the **ricci** daemon and ensure it starts at boot.

```
# service ricci start
Starting oddjobd: [ OK ]
generating SSL certificates... done
Generating NSS database... done
Starting ricci: [ OK ]

# chkconfig ricci on
```

5.4.2 Firewall Configuration

The cluster manager, **corosync** / **cman**, included in the *High Availability* group communicates to each node using UDP ports 5404 and 5405. The distributed lock manager (DLM) communicates using TCP port 21064 and the **ricci** daemon communicates on TCP port 11111. The firewall must be configured to allow traffic between the clustered virtual machines.

Configure the firewall to allow traffic on the ports and save the configuration.

```
# iptables -I INPUT -m state --state NEW -p udp -m multiport --dports
5404,5405 -j ACCEPT

# iptables -I INPUT -m state --state NEW -p tcp -m multiport --dports
11111,21064 -j ACCEPT

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



6 Clustering Virtual Machines Running PostgreSQL

The PostgreSQL database is clustered in an active-passive relationship. Clustering the database in this method requires the following:

- Reliable name resolution for the clustered nodes.
- A dynamic IP address. This address is used for the database service and moves between the cluster members.
- Shared storage. This storage must be available to all cluster nodes since the database files are stored on this storage.
- A properly configured database.
 - Initialize database
 - Configure Firewall
 - Configure SELinux
- A configured cluster.
 - Cluster and node definition
 - Configure fencing
 - PostgreSQL service definition

With the exception of name resolution, the steps needed to configure each node are different. Separate sections are included below for each cluster node and some steps must be completed before others. This entire section of the reference architecture is written in the order the steps should be performed.

6.1 Dynamic IP Address

A dynamic IP address is used to access the PostgreSQL service that is running. This allows the IP address to move between clustered nodes. This allows clients to access a single IP address instead of trying multiple addresses until one succeeds as well as allowing the configured network interfaces to retain their originally configured IP addresses.

The dynamic IP address used to access the PostgreSQL service is 10.16.136.75.

6.2 Clustered Node Name Resolution

The clustered nodes must be able to resolve the network interface names of each node. To ensure name resolution is available at all times, the information is written into the `/etc/hosts` file and the `/etc/nsswitch.conf` file is configured to use the local files for name resolution before using any other method for name resolution. This helps prevent the clustering software from fencing the nodes inappropriately due to name resolution.



The rvmha-db-pri rvmha-db-sec virtual machines have three network interfaces configured. One interface is used for public traffic, one for cluster traffic, and one for iSCSI storage traffic. The interfaces used for clustering and storage are identified by appended **-ci** (cluster interface) or **-si** (storage interface) to the hostname. **Table 6.2.1: PostgreSQL Server Interfaces** contains this information.

Host	Interface	Network	IP Address	Resolvable Name
virtual IP Address	eth0	rhevm	10.16.136.75	rvmha-db-virt
rvmha-db-pri	eth0	rhevm	10.16.136.65	rvmha-db-pri
	eth1	cluster	172.31.0.65	rvmha-db-pri-ci
	eth2	storage	192.168.0.65	rvmha-db-pri-si
rvmha-db-sec	eth0	rhevm	10.16.136.65	rvmha-db-sec
	eth1	cluster	172.31.0.65	rvmha-db-sec-ci
	eth2	storage	192.168.0.65	rvmha-db-sec-si

Table 6.2.1: PostgreSQL Server Interfaces

Append the following to the `/etc/hosts` file on both the rvmha-db-pri and rvmha-db-sec clustered nodes.

`/etc/hosts:`

```
10.16.136.75 rvmha-db-virt
10.16.136.65 rvmha-db-pri
172.31.0.65 rvmha-db-pri-si
192.168.0.65 rvmha-db-pri-ci

10.16.136.66 rvmha-db-sec
172.31.0.66 rvmha-db-sec-si
192.168.0.66 rvmha-db-sec-ci
```

Ensure the `HOSTS` lookup as defined in the `/etc/nsswitch.conf` file has the `FILES` method listed before any other lookup method.

Example line from `/etc/nsswitch.conf`:

```
hosts:      files dns
```



6.3 Primary Node

Perform the steps in this section only on the `rvmha-db-pri` virtual machine. A separate section follows for the `rvmha-db-sec` virtual machine.

6.3.1 Shared Storage

Each of the cluster nodes must be configured to access the storage and present it correctly to the server and cluster software.

The **`iscsi-initiator-utils`** package must be installed. The package contains the daemons and tools necessary to connect to iSCSI targets.

Install the **`iscsi-initiator-utils`** package.

```
# yum -y install iscsi-initiator-utils
Loaded plugins: product-id, rhnplugin, security, subscription-manager
This system is not registered to Red Hat Subscription Management. You can
use subscription-manager to register.
This system is receiving updates from RHN Classic or RHN Satellite.
Setting up Install Process
```

[... Output Abbreviated ...]

```
Installed:
  iscsi-initiator-utils.x86_64 0:6.2.0.873-2.el6

Complete!
```

Servers connecting to iSCSI targets must be uniquely identified. This unique identifier is stored in the `/etc/iscsi/initiatorname.iscsi` file. The **`iscsi-initiator-utils`** package generates a unique identifier and places it in this file upon package installation.

This identifier can be changed to a more meaningful identifier. This is not necessary, but can make reading logs and troubleshooting the iSCSI connections easier.

The identifiers are changed in this environment to aid in identification and troubleshooting if needed. The system serving the iSCSI targets is set to map a LUN to two systems identified as **`iqn.1994-05.com.redhat:rvmhadbpri`** and **`iqn.1994-05.com.redhat:rvmhadbsec`**. Edit the `/etc/iscsi/initiatorname.iscsi` file and change the part of the identifier after the colon (:) to `rvmhadbpri` as seen below.

```
InitiatorName=iqn.1994-05.com.redhat:rvmhadbpri
```



The **iscsiadm** command is used to discover the iSCSI targets that are presented to the virtual machine. The command takes several options to discover the targets.

--mode discoverydb	Place command in discovery mode.
--type sendtargets	Type of discovery to perform.
--portal ip:port	IP address and port of iSCSI server.
--discover	Instructs the command to discover the iSCSI targets.
--print 1	Control how output is displayed. Setting to 1, makes output easier to read. This option is not needed.
--debug 3	Control debug output level. Setting to 3 gives enough information to make troubleshooting connection issues easier. This option is not needed.

Execute the **iscsiadm** command to discover the iSCSI targets.

```
# iscsiadm --mode discoverydb --type sendtargets --portal
172.31.143.202:3260 --discover --print 1 --debug 3
iscsiadm: Max file limits 1024 4096

iscsiadm: Could not open /var/lib/iscsi/send_targets/172.31.143.202,3260: No
such file or directory

iscsiadm: Discovery record [172.31.143.202,3260] not found!
Starting iscsid: [ OK ]
iscsiadm: Could not open /var/lib/iscsi/send_targets/172.31.143.202,3260: No
such file or directory

iscsiadm: starting sendtargets discovery, address 172.31.143.202:3260,
iscsiadm: connecting to 172.31.143.202:3260
iscsiadm: connected local port 49900 to 172.31.143.202:3260
iscsiadm: connected to discovery address 172.31.143.202
iscsiadm: login response status 0000
iscsiadm: discovery process to 172.31.143.202:3260 exiting
iscsiadm: disconnecting conn 0xff5050, fd 3
Target: iqn.1984-
05.com.dell:powervault.md3600i.690b11c00004e0ff000000005061f030
    Portal: 172.31.43.202:3260,1
        Iface Name: default
    Portal: 172.31.143.202:3260,1
        Iface Name: default
    Portal: 172.31.43.203:3260,2
        Iface Name: default
    Portal: 172.31.143.203:3260,2
        Iface Name: default
```

The output shows that the iSCSI target **iqn.1984-05.com.dell:powervault.md3600i.690b11c00004e0ff000000005061f030** is discovered and has four paths available to it.



Once the targets are discovered, they need to be logged into. This allows the operating system to see the targets as disk drives. Using the **iscsiadm** command with the **--mode** option with the argument of **node** and the **--login** option logs into the iSCSI target.

```
# iscsiadm --mode node --login
Logging in to [iface: default, target: iqn.1984-
05.com.dell:powervault.md3600i.690b11c00004e0ff000000005061f030, portal:
172.31.143.203,3260] (multiple)
Logging in to [iface: default, target: iqn.1984-
05.com.dell:powervault.md3600i.690b11c00004e0ff000000005061f030, portal:
172.31.43.203,3260] (multiple)
Logging in to [iface: default, target: iqn.1984-
05.com.dell:powervault.md3600i.690b11c00004e0ff000000005061f030, portal:
172.31.143.202,3260] (multiple)
Logging in to [iface: default, target: iqn.1984-
05.com.dell:powervault.md3600i.690b11c00004e0ff000000005061f030, portal:
172.31.43.202,3260] (multiple)
Login to [iface: default, target: iqn.1984-
05.com.dell:powervault.md3600i.690b11c00004e0ff000000005061f030, portal:
172.31.143.203,3260] successful.
Login to [iface: default, target: iqn.1984-
05.com.dell:powervault.md3600i.690b11c00004e0ff000000005061f030, portal:
172.31.43.203,3260] successful.
Login to [iface: default, target: iqn.1984-
05.com.dell:powervault.md3600i.690b11c00004e0ff000000005061f030, portal:
172.31.143.202,3260] successful.
Login to [iface: default, target: iqn.1984-
05.com.dell:powervault.md3600i.690b11c00004e0ff000000005061f030, portal:
172.31.43.202,3260] successful.
```



After logging into the targets, they are seen in the operating system using the **fdisk -l** command.

```
# fdisk -l
```

```
Disk /dev/vda: 17.2 GB, 17179869184 bytes
16 heads, 63 sectors/track, 33288 cylinders
Units = cylinders of 1008 * 512 = 516096 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00059b07
```

Device	Boot	Start	End	Blocks	Id	System
/dev/vda1	*	3	409	204800	83	Linux

Partition 1 does not end on cylinder boundary.

/dev/vda2		409	2441	1024000	82	Linux swap / Solaris
-----------	--	-----	------	---------	----	----------------------

Partition 2 does not end on cylinder boundary.

/dev/vda3		2441	33289	15547392	8e	Linux LVM
-----------	--	------	-------	----------	----	-----------

Partition 3 does not end on cylinder boundary.

```
Disk /dev/mapper/myvg-rootvol: 15.9 GB, 15904800768 bytes
255 heads, 63 sectors/track, 1933 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
```

```
Disk /dev/sdb: 107.4 GB, 107374182400 bytes
255 heads, 63 sectors/track, 13054 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
```

```
Disk /dev/sda: 107.4 GB, 107374182400 bytes
255 heads, 63 sectors/track, 13054 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
```



The iSCSI appliance used in this reference architecture is a Redundant Disk Array Controller (RDAC) device and requires the **scsi_dh_rdac** device handler module. The module must be loaded before the storage and multipath drivers are loaded. If this module is not loaded, I/O errors can be present in the log files as well as delayed boots.

The `/var/log/messages` log file displays the I/O errors.

```
# tail -5 /var/log/messages
Mar  1 11:43:29 rvmha-db-pri kernel: end_request: I/O error, dev sdb, sector
0
Mar  1 11:43:29 rvmha-db-pri kernel: end_request: I/O error, dev sdd, sector
0
Mar  1 11:43:29 rvmha-db-pri kernel: end_request: I/O error, dev sdb, sector
209715072
Mar  1 11:43:30 rvmha-db-pri kernel: end_request: I/O error, dev sdd, sector
0
Mar  1 11:43:30 rvmha-db-pri kernel: end_request: I/O error, dev sdb, sector
209715072
```

Loading the **scsi_dh_rdac** module is done by appending `rdloaddriver=scsi_dh_rdac` to the end of the kernel line in the `/boot/grub/grub.conf` file.⁴

Backup the `/boot/grub/grub.conf` file then modify it to load the **scsi_dh_rdac** driver on boot.

```
# cp -v /boot/grub/grub.conf{,.orig}
`/boot/grub/grub.conf' -> `/boot/grub/grub.conf.orig'
```

`/boot/grub/grub.conf`:

```
default=0
timeout=5
splashimage=(hd0,0)/grub/splash.xpm.gz
hiddenmenu
title Red Hat Enterprise Linux Server (2.6.32-358.2.1.el6.x86_64)
    root (hd0,0)
    kernel /vmlinuz-2.6.32-358.2.1.el6.x86_64 ro root=/dev/mapper/myvg-
rootvol rd_NO_LUKS LANG=en_US.UTF-8 rd_NO_MD SYSFONT=latarcyrheb-sun16
crashkernel=auto rd_LVM_LV=myvg/rootvol KEYBOARDTYPE=pc KEYTABLE=us
rd_NO_DM rhgb quiet rdloaddriver=scsi_dh_rdac
    initrd /initramfs-2.6.32-358.2.1.el6.x86_64.img
title Red Hat Enterprise Linux (2.6.32-358.el6.x86_64)
    root (hd0,0)
    kernel /vmlinuz-2.6.32-358.el6.x86_64 ro root=/dev/mapper/myvg-
rootvol rd_NO_LUKS LANG=en_US.UTF-8 rd_NO_MD SYSFONT=latarcyrheb-sun16
crashkernel=auto rd_LVM_LV=myvg/rootvol KEYBOARDTYPE=pc KEYTABLE=us
rd_NO_DM rhgb quiet
    initrd /initramfs-2.6.32-358.el6.x86_64.img
```

⁴ See kernel component, BZ#690523 of https://access.redhat.com/knowledge/docs/en-US/Red_Hat_Enterprise_Linux/6/html-single/6.3_Technical_Notes/index.html#kernel_issues



Since multiple paths exist between the cluster node and the iSCSI target, **multipath** is configured to allow the device to fail over between the paths.

Install the **device-mapper-multipath** package.

```
# yum -y install device-mapper-multipath
Loaded plugins: product-id, rhnplugin, security, subscription-manager
This system is not registered to Red Hat Subscription Management. You can
use subscription-manager to register.
This system is receiving updates from RHN Classic or RHN Satellite.
```

[... Output Abbreviated ...]

Installed:

device-mapper-multipath.x86_64 0:0.4.9-64.el6

Dependency Installed:

device-mapper-multipath-libs.x86_64 0:0.4.9-64.el6

Complete!

The configuration for the **multipath** daemon is stored in the `/etc/multipath.conf` file. The multipath configuration being used in this reference architecture is below. For more information on how this configuration was obtained, see the manual page for **multipathd**, `multipath.conf`, and the iSCSI target device product documentation.⁵

Modify the `/etc/multipath.conf` file for use with the iSCSI storage array.

```
defaults {
    find_multipaths yes
    user_friendly_names yes
}

blacklist {
}

devices {
    device {
        vendor "DELL"
        product "MD36xxi"
        path_grouping_policy group_by_prio
        prio rdac
        path_checker rdac
        path_selector "round-robin 0"
        hardware_handler "1 rdac"
        failback immediate
        features "2 pg_init_retries 50"
        no_path_retry 30
        rr_min_io_rq 100
    }
}
```

⁵ <http://www.dell.com/downloads/global/products/pvaul/en/powervault-md36x0i-linux-dm-installation.pdf>



Start the **multipathd** service and configure it to start upon boot.

```
# service multipathd start
Starting multipathd daemon: [ OK ]

# chkconfig multipathd on
```

View the paths by issuing the **multipath -l** command.

```
# multipath -l
mpatha (3690b11c00004e0ff0000030950e7b62f) dm-1 DELL,MD36xxi
size=100G features='3 queue_if_no_path pg_init_retries 50' hwhandler='1
rdac' wp=rw
|+- policy='round-robin 0' prio=0 status=active
| |- 5:0:0:8 sda 8:0 active undef running
| `-- 4:0:0:8 sdc 8:32 active undef running
`+- policy='round-robin 0' prio=0 status=enabled
| - 3:0:0:8 sdd 8:48 active undef running
`- 2:0:0:8 sdb 8:16 active undef running
```

Listing the device paths, displays the World Wide Identifier (WWID) for the iSCSI target after the device name. A more human readable name can be given to the iSCSI target by adding an alias definition in the */etc/multipath.conf* file. Using the WWID, modify the */etc/multipath.conf* file to assign an alias of *db_share* to the iSCSI target.

```
multipaths {
    multipath {
        wwid 3690b11c00004e0ff0000030950e7b62f
        alias db_share
    }
}
```

The configuration file for **multipathd** service must be reloaded before **multipathd** can use the newly created alias. Reload the configuration file using the **service reload** command.

```
# service multipathd reload
Reloading multipathd: [ OK ]
```

The **multipath -l** command now displays the alias name.

```
# multipath -l
db_share (3690b11c00004e0ff0000030950e7b62f) dm-1 DELL,MD36xxi
size=100G features='3 queue_if_no_path pg_init_retries 50' hwhandler='1
rdac' wp=rw
|+- policy='round-robin 0' prio=0 status=active
| |- 5:0:0:8 sda 8:0 active undef running
| `-- 4:0:0:8 sdc 8:32 active undef running
`+- policy='round-robin 0' prio=0 status=enabled
| - 3:0:0:8 sdd 8:48 active undef running
`- 2:0:0:8 sdb 8:16 active undef running
```



The path to access the device is now `/dev/mapper/db_share`. The device must be partitioned and a file system created on the partition before the operating system can use the device to store the database files.

The **parted** command is used to create a new partition and partition table on the device. The **parted** command **mkpart** is used to create a primary partition that starts on the first disk location (**0**) and ends at the last location (**-1s**). The **--align cylinder** option is used to prevent the **parted** command from reported a misaligned partition.

```
# parted /dev/mapper/db_share mklabel msdos
Information: You may need to update /etc/fstab.

# parted --align cylinder /dev/mapper/db_share -- mkpart primary 0 -1s
Information: You may need to update /etc/fstab.
```

Create an ext4 file system on the new partition. A label is placed on the file system to allow the file system to be accessed using a more readable name. The file system is given a label of `SHARE_DB` by passing the **-L** argument to the command.

```
# mke2fs -t ext4 -L SHARE_DB /dev/mapper/db_sharep1
mke2fs 1.41.12 (17-May-2010)
Filesystem label=SHARE_DB
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
6553600 inodes, 26214399 blocks
1310719 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=4294967296
800 block groups
32768 blocks per group, 32768 fragments per group
8192 inodes per group
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,
    4096000, 7962624, 11239424, 20480000, 23887872

Writing inode tables: done
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 36 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
```



6.3.2 Database Initialization

The daemon and commands needed to run and access a PostgreSQL database are in the **postgresql** and **postgresql-server** packages. Use yum to install these packages.

```
# yum -y install postgresql postgresql-server
Loaded plugins: product-id, rhnplugin, security, subscription-manager
This system is not registered to Red Hat Subscription Management. You can
use subscription-manager to register.
This system is receiving updates from RHN Classic or RHN Satellite.
```

[... Output Abbreviated ...]

```
Installed:
  postgresql.x86_64 0:8.4.13-1.el6_3  postgresql-server.x86_64 0:8.4.13-
1.el6_3
```

```
Dependency Installed:
  postgresql-libs.x86_64 0:8.4.13-1.el6_3
```

```
Complete!
```

The database files are stored on the shared storage that was created earlier. The partition is mounted on the `/var/share_db` directory. Since this directory does not exist, it must be created.

```
# mkdir /var/share_db
```

The newly created file system must be mounted in order to fix permission issues and to initialize the database. Mount the file system.

```
# mount LABEL=SHARE_DB /var/share_db
```

Since the database runs as the *postgres* user, the owner and group of the directory must be set to *postgres*. This is needed to allow **postgres** to write to the directory and to the files within it.

Change the user and group ownership's of the `/var/share_db` directory.

```
# chown -v postgres.postgres /var/share_db
```

```
changed ownership of `/var/share_db' to postgres:postgres
```



A new database must be initialized before the PostgreSQL server can use it. Since the database runs as the *postgres* user, this step is performed as the *postgres* user. This ensures the ownership of the files and directories created are set correctly.

The **sudo** command is used to call the **initdb** command to initialize the database as the *postgres* user. The **-D** option is used to specify which directory to create the database in.

```
# sudo -u postgres initdb -D /var/share_db/data
could not change directory to "/root"
The files belonging to this database system will be owned by user
"postgres".
This user must also own the server process.

The database cluster will be initialized with locale en_US.UTF-8.
The default database encoding has accordingly been set to UTF8.
The default text search configuration will be set to "english".

creating directory /var/share_db/data ... ok
creating subdirectories ... ok
selecting default max_connections ... 100
selecting default shared_buffers ... 32MB
creating configuration files ... ok
creating template1 database in /var/share_db/data/base/1 ... ok
initializing pg_authid ... ok
initializing dependencies ... ok
creating system views ... ok
loading system objects' descriptions ... ok
creating conversions ... ok
creating dictionaries ... ok
setting privileges on built-in objects ... ok
creating information schema ... ok
vacuuming database template1 ... ok
copying template1 to template0 ... ok
copying template1 to postgres ... ok

WARNING: enabling "trust" authentication for local connections
You can change this by editing pg_hba.conf or using the -A option the
next time you run initdb.

Success. You can now start the database server using:

    postgres -D /var/share_db/data
or
    pg_ctl -D /var/share_db/data -l logfile start
```



6.3.3 Firewall Configuration

The default port used by the PostgreSQL server is TCP port 5432. The firewall must be configured to allow the network traffic to come through this port. Use the **iptables** command to allow incoming traffic on the port and then save the configuration.

```
# iptables -I INPUT -m state --state NEW -m tcp -p tcp --dport 5432 -j ACCEPT

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

6.3.4 SELinux

SELinux⁶ prevents the database from functioning correctly since the files are in a non-standard location on the file system. A new SELinux policy must be created to set the correct security contexts on the files in the `/var/share_db` directory and sub-directories.

The **semanage** command is used to create new SELinux policies and is available in the **policycoreutils-python** package. Install the **policycoreutils-python** package.

```
# yum -y install policycoreutils-python
Loaded plugins: product-id, rhnplugin, security, subscription-manager
This system is not registered to Red Hat Subscription Management. You can
use subscription-manager to register.
This system is receiving updates from RHN Classic or RHN Satellite.

[ ... Output Abbreviated ... ]

Installed:
  policycoreutils-python.x86_64 0:2.0.83-19.30.el6

Complete!
```

The **semanage** command is executed to add a new file context labeling entry for the `/var/share_db` directory, its sub-directories, and files. This new entry assigns the `POSTGRESQL_DB_T` SELinux type to the files and directories.

The **-a** option indicates that an *ADD* action is to be taken by the command. The **-t** option specifies the SELinux security *TYPE* to for the object being added, the database directory needs the `POSTGRESQL_DB_T` object type defined. The final argument to the **semanage** command is a regular expression that matches a directory, sub-directories, and contents.

```
# semanage fcontext -a -t postgresql_db_t "/var/share_db(/.*)?"
```

6 https://access.redhat.com/site/documentation/en-US/Red_Hat_Enterprise_Linux/4/html/SELinux_Guide/selg-preface-0011.html



After the new entry is created, the security contexts on the database directory must be set correctly. The **restorecon** command is executed to relabel the directory and files with the correct security context.

```
# restorecon -R -v /var/share_db
restorecon reset /var/share_db context system_u:object_r:file_t:s0-
>system_u:object_r:postgresql_db_t:s0
restorecon reset /var/share_db/lost+found context
system_u:object_r:file_t:s0->system_u:object_r:postgresql_db_t:s0

[ ... Output Abbreviated ... ]

restorecon reset /var/share_db/data/global/1136_fsm context
unconfined_u:object_r:file_t:s0->unconfined_u:object_r:postgresql_db_t:s0
restorecon reset /var/share_db/data/global/1136 context
unconfined_u:object_r:file_t:s0->unconfined_u:object_r:postgresql_db_t:s0
restorecon reset /var/share_db/data/global/2671 context
unconfined_u:object_r:file_t:s0->unconfined_u:object_r:postgresql_db_t:s0
```

The file system should not be mounted when the cluster is active unless the cluster node is hosting the PostgreSQL database. Unmount the file system.

```
# umount /var/share_db
```

The cluster software contains a service definition that controls the PostgreSQL database. The service copies the original *postgres.conf* file, potentially makes changes to the copy, and uses the copy to start the database. The SELinux file context on the copied configuration file is *CLUSTER_CONF_T* and not *POSTGRESQL_DB_T*. This label change prevents the PostgreSQL database from starting because SELinux prevents the PostgreSQL daemon from reading the configuration file.

This is a known issue and a bugzilla bug⁷ is opened on it. A fix has been made and should be included in a future release of the SELinux policies. Until the bug fix is included in a policy release, a custom policy must be created to allow the PostgreSQL daemon to read the configuration file.

⁷ https://bugzilla.redhat.com/show_bug.cgi?id=814334



The following is the policy that fixes the issue. Create a file under the `/root` directory called `mypol.te` that contains the policy definition.

`/root/mypol.te:`

```
policy_module(mypol,1.0)

require{
    type rgmanager_t;
    type postgresql_t;
    type cluster_conf_t;
    type rgmanager_var_run_t;
}

read_files_pattern(postgresql_t, cluster_conf_t, cluster_conf_t)
manage_files_pattern(postgresql_t, rgmanager_var_run_t, rgmanager_var_run_t)
```

Compile the new policy so it can be inserted into the system policies.

```
# make -f /usr/share/selinux/devel/Makefile mypol.pp
Compiling targeted mypol module
/usr/bin/checkmodule: loading policy configuration from tmp/mypol.tmp
/usr/bin/checkmodule: policy configuration loaded
/usr/bin/checkmodule: writing binary representation (version 10) to
tmp/mypol.mod
Creating targeted mypol.pp policy package
rm tmp/mypol.mod.fc tmp/mypol.mod
```

Install the newly compiled policy module into the SELinux infrastructure.

```
# semodule -i mypol.pp
```

6.3.5 Cluster Configuration

Configuring the cluster software is performed on a single node in the cluster and then pushed to the remaining cluster nodes. Various methods exist to configure the cluster including a GUI configuration tool, a command line tool, and directly editing the configuration files. This reference architecture uses the **ccs** command line tool, provided by the **ccs** package, to configure the cluster.

When invoking the **ccs** command, a host must be specified. This is because the **ccs** command communicates to the **ricci** daemon and the **ricci** daemon writes the changes to the configuration file. The **--host** option is used to specify the host.

The interface with an IP address on the 192.168.0/24 network is used for cluster communication. The interface of each machine that is on the 192.168.0/24 network has the resolvable host name of **rvmha-db-pri-ci** and **rvmha-db-sec-ci** respectively.



6.3.5.1 Define Cluster and Add Nodes

The first step in the configuration is to define a cluster. The **--createcluster** option is used. Specify a cluster name of **rmha-db** to create the cluster.

```
# ccs --host rmha-db-pri-ci --createcluster rmha-db  
rmha-db-pri-ci password: [Password]
```

Since the cluster in this reference architecture is a two node cluster, some special configuration must be enabled in order to prevent each node from losing quorum and fencing the other. This is known as a spit-brain situation. The **--setman** option with the **two_node=1** and **expected_votes=1** arguments alleviates a split-brain situation.

```
# ccs -h rmha-db-pri-ci --setman two_node=1 expected_votes=1
```

After the cluster is created, the nodes must be added to it. The **--addnode** option performs this action. Specify the **rmha-db-pri-ci** and **rmha-db-sec-ci** interfaces as options when invoking the command. This ensures that cluster communication is performed on the correct network and not mixed with the public network.

```
# ccs --host rmha-db-pri-ci --addnode rmha-db-pri-ci  
Node rmha-db-pri-ci added.  
  
# ccs --host rmha-db-pri-ci --addnode rmha-db-sec-ci  
Node rmha-db-sec-ci added.
```

6.3.5.2 Configure Fencing

Fencing is a method that allows one node to prevent the other from accessing resources when it should not. When one node stops responding, the second node can fence the node and prevent it from starting the services.

Since the nodes are virtual machines, the **fence_rhevm** fencing method is used. This allows the cluster nodes to power off each other if need be.

A fence device must be created. This device is used to fence each cluster node. The **--addfencedev** option is used to create a fence device. It is followed by a name for the fence device, this reference architecture uses **RHEVM-API** as the device name. The **agent** option specifies the fencing agent to use. For fencing virtual machines in a Red Hat Enterprise Virtualization environment, the **fence_rhevm** agent is used.

This agent takes several arguments which must be specified and a few that are optional. The **ipaddr**, **login**, and **passwd** arguments must be specified. These arguments specify how to connect to the REST API of the virtualization environment. The **login** and **passwd** arguments are credentials that can be used to log into the Red Hat Enterprise Virtualization Manager. The **ssl** argument is optional and indicates that communications to the REST API are to be done using the **ssl** transport.



Add the fence device to the cluster configuration.

```
# ccs --host rvmha-db-pri-ci --addfencedev RHEVM-API agent=fence_rhevm  
ipaddr=jh-rhevm.cloud.lab.eng.bos.redhat.com login="[User]"  
passwd="[Password]" ssl=1
```

Check to ensure the fencing device was created using the **ccs** command with the **--lsfencedev** option.

```
# ccs --host rvmha-db-pri-ci --lsfencedev  
  
RHEVM-API: passwd=100yard-, ipaddr=jh-rhevm.cloud.lab.eng.bos.redhat.com,  
agent=fence_rhevm, ssl=1, login=admin
```

Each cluster node needs a method of fencing specified for it. The method names may be the same or different for each device. The **--addmethod** option requires that the first argument passed to it is the name of the method. The second argument passed is the host to add the method to.

Create a fencing method called **API** for each cluster node. The name **API** is chosen as a meaningful name since the fencing method will use the REST API interface into the Red Hat Enterprise Virtualization Manager to control the device.

```
# ccs --host rvmha-db-pri-ci --addmethod API rvmha-db-pri-ci  
  
Method API added to rvmha-db-pri-ci.
```

```
# ccs --host rvmha-db-pri-ci --addmethod API rvmha-db-sec-ci  
  
Method API added to rvmha-db-sec-ci.
```

The final step for fencing configuration is adding a fence instance. The fence instance ties the fence device to the fence method and cluster node. The **--addfenceinst** option specifies the fence instance and takes four arguments. The first argument is the name of the fence device to use, the second argument is the host for which the instance applies. The third argument is the fencing method to use. The fourth argument specifies the port of the virtual machine. The port specified is actually the virtual machine name as it is displayed in the Red Hat Enterprise Virtualization Manager.

Add a fencing instance for each cluster node.

```
# ccs --host rvmha-db-pri-ci --addfenceinst RHEVM-API rvmha-db-pri-ci API  
port=rvmha-db-pri  
  
# ccs --host rvmha-db-pri-ci --addfenceinst RHEVM-API rvmha-db-sec-ci API  
port=rvmha-db-sec
```



Check that the fencing instances are created correctly. Use the **ccs** command with the **--lsfenceinst**.

```
# ccs --host rvmha-db-pri-ci --lsfenceinst

rvmha-db-pri-ci
API
  RHEVM-API: port=rvmha-db-pri
rvmha-db-sec-ci
API
  RHEVM-API: port=rvmha-db-sec
```

6.3.5.3 Configure a Failover Domain

The virtual machines acting as cluster nodes are pinned to particular Red Hat Enterprise Virtualization Hypervisor hosts to aid in high availability and load balancing. The services defined should run on the primary cluster node when possible. Normally when a service fails over, it does not automatically fail back to the original node unless manual intervention is performed. The desirable effect is to have the service relocate to the primary node when it becomes available.

Assigning a single cluster node and the service to a failover domain achieves the desired effect. When the primary node fails, the cluster service is allowed to run on any node that is not in the failover domain. But as soon as a member of the failover domain is available, the service relocates to a member of the failover domain. Since there is only one member, the service relocates to the primary cluster node.

The **--addfailoverdomain** option is used to create a failover domain. This option can take several arguments. The first argument must be the name of the failover domain to create. The remaining arguments specify behaviors of the failover domain. The **unrestricted** argument tells the cluster service to allow the service to run on and cluster node. The **ordered** argument instructs the cluster software to allow priority ordering for cluster members in a failover domain. The **failback** argument tells the cluster software to relocate the service to the failover domain member that has the highest priority.

Add a failover domain called **database_dom** to the cluster configuration.

```
# ccs --host rvmha-db-pri-ci --addfailoverdomain database_dom unrestricted
ordered failback
```



A node must be added to the failover domain. This is done using the **--addfailoverdomain** option. This option takes several arguments. The first argument is the name of the failover domain to add the node to. The second argument is the cluster node to add to the failover domain and the third argument indicates the priority of the cluster node.

Add the primary cluster node to the failover domain with a priority of **1**.

```
# ccs --host rvmha-db-pri-ci --addfailoverdomainnode database_dom rvmha-db-pri-ci 1
```

Verify the failover domain is created correctly using the **--lsfailoverdomain** option.

```
# ccs --host rvmha-db-pri-ci --lsfailoverdomain

database_dom: restricted=0, ordered=1, nofailback=0
rvmha-db-pri-ci: priority=1
```

6.3.5.4 Define PostgreSQL Service

The final step in creating the cluster configuration is creating the service definition for the PostgreSQL daemon. Defining the daemon requires that all the resources needed by the service are available. The PostgreSQL service requires that the directory the database files are stored in are available, this means the shared storage must be mounted in the appropriate place. It also requires that the virtual IP address is available on the cluster node running the service. And finally the PostgreSQL daemon must be started.

Keeping the requirements for the PostgreSQL service in mind, a cluster service must include the following:

- Virtual IP address available
- Shared storage mounted
- PostgreSQL started

The first step in service creation is to define the service. This is done using the **--addservice** option. The first argument to the option is the name of the service. The remaining arguments are key-value pairs and are optional if not needed. The **domain** key specifies the failover domain that the service belongs to. The **autostart** key indicates that this service should automatically start when the cluster software is started. The **recovery** key identifies the type of recovery to be done on the service, the **relocate** value tells the cluster daemon to relocate the service to another cluster node.

Create a cluster service with the name of **postgres**.

```
# ccs --host rvmha-db-pri-ci --addservice postgres domain=database_dom autostart=1 recovery=relocate
```



Once the service is defined, sub-services or resource requirements can be added. The **--addsubservice** option is used to add sub-services. The first argument is the name of the service to add the sub-service to. The second option is the type of sub-service. The remaining arguments are options that are unique to each sub-service type.

Add a sub-service with the type of **ip**. Specify the **address** key with a value of **10.16.136.75**. This IP address is the virtual IP address for the PostgreSQL service.

```
# ccs --host rvmha-db-pri-ci --addsubservice postgres ip
address=10.16.136.75
```

The shared storage must be mounted for the service to operate. It uses a sub-service type of **fs**. Ensuring the IP address is available before the shared storage is mounted can be done by telling the cluster software the **ip** sub-service type is required before the shared storage can be mounted. This is done by prepending the sub-service type with the required resource type separated with a colon. For example **ip:fs** indicates the **ip** type must be available before the **fs** type can be made available.

Add the shared storage definition, ensuring the IP address is available before making the **fs** resource available. The **fs** sub-service type takes several key-value pairs. The **name** key specifies the name of the resource. The **mountpoint** key specifies where to mount the storage. The **device** key specifies the device to mount. Device labels can be used here. The **force_fsck** option indicates if a file system check should be performed prior to mounting the device.

```
# ccs --host rvmha-db-pri-ci --addsubservice postgres ip:fs
name=db_shared_fs mountpoint=/var/share_db device=LABEL=SHARE_DB
force_fsck=1
```

The cluster software contains a service method specifically for the PostgreSQL database. This service method is called **postgres-8** and takes several key-value pairs as options. The **name** key specifies a name for the resource. The **config_file** key specifies the location of the database configuration file. The **postmaster_options** key specifies any options that are passed to the PostgreSQL daemon upon start.

Add a sub-service definition for the **postgres-8** service type. Make sure the **ip** and **fs** resources are available before starting the service. Specify a name of **postgres_db**. Specify a configuration file of **/var/share_db/data/postgresql.conf**. And specify the **postmaster_options** of **-D /var/share_db/data**. Specifying the **postmaster_options** key is required since the database is in a non-standard location.

```
# ccs --host rvmha-db-pri-ci --addsubservice postgres ip:fs:postgres-8
name=postgres_db config_file=/var/share_db/data/postgresql.conf
postmaster_options="-D /var/share_db/data"
```



Use the **--lsservices** option to display the newly created service definition.

```
# ccs --host rvmha-db-pri-ci --lsservices
service: name=postgres, domain=database_dom, autostart=1, recovery=relocate
  ip: address=10.16.136.75
    fs: name=db_shared_fs, device=LABEL=SHARE_DB, mountpoint=/var/share_db,
force_fsck=1
  postgres-8: name=postgres_db,
config_file=/var/share_db/data/postgresql.conf, postmaster_options=-D
/var/share_db/data
resources:
```

6.4 Secondary Node

The steps to configure node two are listed in this section. They are a sub-set of the steps used to configure the primary node and are not explained here for brevity. For an explanation of the steps, please refer to the primary node section.

6.4.1 Storage

Install the **iscsi-initiator-utils** package.

```
# yum -y install iscsi-initiator-utils
Loaded plugins: product-id, rhnplugin, security, subscription-manager
This system is not registered to Red Hat Subscription Management. You can
use subscription-manager to register.
This system is receiving updates from RHN Classic or RHN Satellite.
Setting up Install Process

[ ... Output Abbreviated ... ]

Installed:
  iscsi-initiator-utils.x86_64 0:6.2.0.873-2.el6

Complete!
```

Edit the `/etc/iscsi/initiatorname.iscsi` file and change the part of the identifier after the colon (:) to `rvmhadbsec` as seen below.

```
InitiatorName=iqn.1994-05.com.redhat:rvmhadbsec
```



Execute the **iscsiadm** command to discover the iSCSI targets.

```
# iscsiadm --mode discoverydb --type sendtargets --portal
172.31.143.202:3260 --discover --print 1 --debug 3
iscsiadm: Max file limits 1024 4096

iscsiadm: Could not open /var/lib/iscsi/send_targets/172.31.143.202,3260: No
such file or directory

iscsiadm: Discovery record [172.31.143.202,3260] not found!
Starting iscsid: [ OK ]
iscsiadm: Could not open /var/lib/iscsi/send_targets/172.31.143.202,3260: No
such file or directory

iscsiadm: starting sendtargets discovery, address 172.31.143.202:3260,
iscsiadm: connecting to 172.31.143.202:3260
iscsiadm: connected local port 49900 to 172.31.143.202:3260
iscsiadm: connected to discovery address 172.31.143.202
iscsiadm: login response status 0000
iscsiadm: discovery process to 172.31.143.202:3260 exiting
iscsiadm: disconnecting conn 0xff5050, fd 3
Target: iqn.1984-
05.com.dell:powervault.md3600i.690b11c00004e0ff000000005061f030
    Portal: 172.31.43.202:3260,1
        Iface Name: default
    Portal: 172.31.143.202:3260,1
        Iface Name: default
    Portal: 172.31.43.203:3260,2
        Iface Name: default
    Portal: 172.31.143.203:3260,2
        Iface Name: default
```



Login to the iscsi targets.

```
# iscsiadm --mode node --login
Logging in to [iface: default, target: iqn.1984-
05.com.dell:powervault.md3600i.690b11c00004e0ff000000005061f030, portal:
172.31.143.203,3260] (multiple)
Logging in to [iface: default, target: iqn.1984-
05.com.dell:powervault.md3600i.690b11c00004e0ff000000005061f030, portal:
172.31.43.203,3260] (multiple)
Logging in to [iface: default, target: iqn.1984-
05.com.dell:powervault.md3600i.690b11c00004e0ff000000005061f030, portal:
172.31.143.202,3260] (multiple)
Logging in to [iface: default, target: iqn.1984-
05.com.dell:powervault.md3600i.690b11c00004e0ff000000005061f030, portal:
172.31.43.202,3260] (multiple)
Login to [iface: default, target: iqn.1984-
05.com.dell:powervault.md3600i.690b11c00004e0ff000000005061f030, portal:
172.31.143.203,3260] successful.
Login to [iface: default, target: iqn.1984-
05.com.dell:powervault.md3600i.690b11c00004e0ff000000005061f030, portal:
172.31.43.203,3260] successful.
Login to [iface: default, target: iqn.1984-
05.com.dell:powervault.md3600i.690b11c00004e0ff000000005061f030, portal:
172.31.143.202,3260] successful.
Login to [iface: default, target: iqn.1984-
05.com.dell:powervault.md3600i.690b11c00004e0ff000000005061f030, portal:
172.31.43.202,3260] successful.
```

Modify the `/boot/grub/grub.conf` file to ensure the system loads the **scsi_dh_rdac** driver upon boot.

```
default=0
timeout=5
splashimage=(hd0,0)/grub/splash.xpm.gz
hiddenmenu
title Red Hat Enterprise Linux Server (2.6.32-358.2.1.el6.x86_64)
    root (hd0,0)
    kernel /vmlinuz-2.6.32-358.2.1.el6.x86_64 ro root=/dev/mapper/myvg-
rootvol rd_NO_LUKS LANG=en_US.UTF-8 rd_NO_MD SYSFONT=latarcyrheb-sun16
crashkernel=auto rd_LVM_LV=myvg/rootvol KEYBOARDTYPE=pc KEYTABLE=us
rd_NO_DM rhgb quiet rdloaddriver=scsi_dh_rdac
    initrd /initramfs-2.6.32-358.2.1.el6.x86_64.img
title Red Hat Enterprise Linux (2.6.32-358.el6.x86_64)
    root (hd0,0)
    kernel /vmlinuz-2.6.32-358.el6.x86_64 ro root=/dev/mapper/myvg-
rootvol rd_NO_LUKS LANG=en_US.UTF-8 rd_NO_MD SYSFONT=latarcyrheb-sun16
crashkernel=auto rd_LVM_LV=myvg/rootvol KEYBOARDTYPE=pc KEYTABLE=us
rd_NO_DM rhgb quiet
    initrd /initramfs-2.6.32-358.el6.x86_64.img
```




Install the **device-mapper-multipath** package.

```
# yum -y install device-mapper-multipath
Loaded plugins: product-id, rhnplugin, security, subscription-manager
This system is not registered to Red Hat Subscription Management. You can
use subscription-manager to register.
This system is receiving updates from RHN Classic or RHN Satellite.
```

[... Output Abbreviated ...]

Installed:

device-mapper-multipath.x86_64 0:0.4.9-64.el6

Dependency Installed:

device-mapper-multipath-libs.x86_64 0:0.4.9-64.el6

Complete!



Modify the `/etc/multipath.conf` file for use with the iSCSI storage array. The WWID of the iSCSI LUN is the same as it was for the primary node.

```
defaults {
    find_multipaths yes
    user_friendly_names yes
}

blacklist {
}

multipaths {
    multipath {
        wwid 3690b11c00004e0ff0000030950e7b62f
        alias db_share
    }
}

devices {
    device {
        vendor "DELL"
        product "MD36xxi"
        path_grouping_policy group_by_prio
        prio rdac
        path_checker rdac
        path_selector "round-robin 0"
        hardware_handler "1 rdac"
        failback immediate
        features "2 pg_init_retries 50"
        no_path_retry 30
        rr_min_io_rq 100
    }
}
```

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

```
# service multipathd start
Starting multipathd daemon: [ OK ]

# chkconfig multipathd on
```

The **multipath -l** command now displays the alias name.

```
# multipath -l
db_share (3690b11c00004e0ff0000030950e7b62f) dm-1 DELL,MD36xxi
size=100G features='3 queue_if_no_path pg_init_retries 50' hwhandler='1
rdac' wp=rw
|+- policy='round-robin 0' prio=0 status=active
| |- 5:0:0:8 sda 8:0 active undef running
| `-- 4:0:0:8 sdc 8:32 active undef running
`+- policy='round-robin 0' prio=0 status=enabled
| - 3:0:0:8 sdd 8:48 active undef running
`- 2:0:0:8 sdb 8:16 active undef running
```



6.4.2 Database Initialization

Install the **postgresql** and **postgresql-server** packages using **yum**.

```
# yum -y install postgresql postgresql-server
Loaded plugins: product-id, rhnplugin, security, subscription-manager
This system is not registered to Red Hat Subscription Management. You can
use subscription-manager to register.
This system is receiving updates from RHN Classic or RHN Satellite.

[ ... Output Abbreviated ... ]

Installed:
  postgresql.x86_64 0:8.4.13-1.el6_3  postgresql-server.x86_64 0:8.4.13-
1.el6_3

Dependency Installed:
  postgresql-libs.x86_64 0:8.4.13-1.el6_3

Complete!
```

Create the **/var/share_db** directory.

```
# mkdir /var/share_db
```

6.4.3 Firewall Configuration

Open the firewall for TCP port 5432 and save the configuration.

```
# iptables -I INPUT -m state --state NEW -m tcp -p tcp --dport 5432 -j
ACCEPT

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

6.4.4 SELinux

Install the **policycoreutils-python** package.

```
# yum -y install policycoreutils-python
Loaded plugins: product-id, rhnplugin, security, subscription-manager
This system is not registered to Red Hat Subscription Management. You can
use subscription-manager to register.
This system is receiving updates from RHN Classic or RHN Satellite.

[ ... Output Abbreviated ... ]

Installed:
  policycoreutils-python.x86_64 0:2.0.83-19.30.el6

Complete!
```



Add a new file context labeling entry for `/var/share_db` directory to SELinux.

```
# semanage fcontext -a -t postgresql_db_t "/var/share_db(/.*)?"
```

Create the SELinux policy that allows the PostgreSQL daemon to read the configuration file created by the cluster software.

```
# scp rvmha-db-pri:/root/mypol.te .
root@rvmha-db-pri's password: [Password]
mypol.te                               100% 267      0.3KB/s   00:00
```

Compile the new policy.

```
# make -f /usr/share/selinux/devel/Makefile mypol.pp
Compiling targeted mypol module
/usr/bin/checkmodule: loading policy configuration from tmp/mypol.tmp
/usr/bin/checkmodule: policy configuration loaded
/usr/bin/checkmodule: writing binary representation (version 10) to
tmp/mypol.mod
Creating targeted mypol.pp policy package
rm tmp/mypol.mod.fc tmp/mypol.mod
```

Install the newly compiled policy module into the SELinux infrastructure.

```
# semodule -i mypol.pp
```

6.5 Starting the Cluster

On the primary node, issue the **ccs_sync** command to propagate the *cluster.conf* file to the secondary node.

```
# ccs_sync
You have not authenticated to the ricci daemon on rvmha-db-pri-ci
Password: [Password]
You have not authenticated to the ricci daemon on rvmha-db-sec-ci
Password: [Password]
```

Use the **ccs** command to start the cluster services on all the cluster nodes. The **--startall** option instructs the cluster services to start and also enables the services to start at boot.

```
# ccs --host rvmha-db-pri-ci --startall

rvmha-db-sec-ci password: [Password]
Started rvmha-db-sec-ci
Started rvmha-db-pri-ci
```



Use the **clustat** command to see the status of the cluster and its services.

```
# clustat
Cluster Status for rvmha-db @ Thu Mar 14 11:15:26 2013
Member Status: Quorate

Member Name                ID    Status
-----
rvmha-db-pri-ci            1 Online, Local, rgmanager
rvmha-db-sec-ci            2 Online, rgmanager

Service Name                Owner (Last)                State
-----
service:postgres            rvmha-db-pri-ci            started
```

Now that clustering is up, test to make sure the service can stat on both nodes. The **clussvcadm** command is used to control the cluster services. The **-r** option tells the command to relocate a service. The **-m** option specifies which node member to start the service on.

```
# clussvcadm -r postgres -m rvmha-db-sec-ci
Trying to relocate service:postgres to rvmha-db-sec-ci...Success
service:postgres is now running on rvmha-db-sec-ci

# clustat
Cluster Status for rvmha-db @ Thu Apr 11 11:06:45 2013
Member Status: Quorate

Member Name                ID    Status
-----
rvmha-db-pri-ci            1 Online, Local, rgmanager
rvmha-db-sec-ci            2 Online, rgmanager

Service Name                Owner (Last)                State
-----
service:postgres            rvmha-db-sec-ci            started
```



Make sure the service can be restarted on the primary node.

```
# clusvcadm -r postgres -m rvmha-db-pri-ci
Trying to relocate service:postgres to rvmha-db-pri-ci...Success
service:postgres is now running on rvmha-db-pri-ci

# clustat
Cluster Status for rvmha-db @ Thu Apr 11 11:07:29 2013
Member Status: Quorate

Member Name                                ID    Status
-----
rvmha-db-pri-ci                            1 Online, Local, rgmanager
rvmha-db-sec-ci                            2 Online, rgmanager

Service Name                                Owner (Last)                                State
-----
service:postgres                            rvmha-db-pri-ci                            started
```

The cluster nodes are not yet using the **scsi_dh_rdac** drivers. The system needs to be rebooted to ensure the drivers are loaded. Rebooting the system also tests that the cluster services start correctly at boot.

Reboot both nodes by issuing the **init 6** command on the rvmha-db-pri primary node. It may take a few minutes for the primary node to reboot since it relocate the **postgres** service to the secondary node.

```
# init 6
```

Once the primary node is booted, it should fence the secondary. This causes the **postgres** service to start on the primary node and the secondary node to restart.



7 Clustering Virtual Machines Running a rsyslog Server

The `rsyslog` service can be configured to act as a central log server for various systems or devices. It can also be configured to log entries to a database. This allows a central repository for all log entries that can be easily parsed and reported on.

Clustering the `rsyslog` service is different than the PostgreSQL database. Since it is desirable to log entries from all systems as well as the cluster nodes themselves, both systems must always run the `rsyslog` service.

The `rsyslog` service on each cluster node is configured to accept log entries from other systems and to also write these entries to the database. A virtual IP address is configured as a clustered service so it can move between the clustered nodes. This virtual IP address is used by the clients when connecting to the logging server.

Clustering the `rsyslog` service requires the following:

- Reliable name resolution for the clustered nodes.
- A dynamic IP address. This address is used for the `rsyslog` incoming connections and moves between the cluster members.
- A properly configured `rsyslog` daemon.
 - Enable logging to a database
 - Allow remote logs to be received
- Configure SELinux
- A created and configured PostgreSQL database. The database and tables must be created before use.
- A configured cluster.
 - Cluster and node definition
 - Configure fencing
 - IP address service definition

As with clustering the PostgreSQL database, the steps required to configure each node are different. Separate sections for each node are listed below. For brevity, some steps are not explained since they were discussed in the **Clustering Virtual Machines Running PostgreSQL** section.



7.1 Clustered Node Name Resolution

Table 7.1.1: rsyslog Server Interfaces contains the interface configuration for the rsyslog cluster nodes.

Host	Interface	Network	IP Address	Resolvable Name
virtual IP Address	eth0	rhevm	10.16.136.74	rvmha-log-virt
rvmha-log-pri	eth0	rhevm	10.16.136.67	rvmha-log-pri
	eth1	cluster	192.168.0.67	rvmha-log-pri-ci
rvmha-log-sec	eth0	rhevm	10.16.136.68	rvmha-log-sec
	eth1	cluster	192.168.0.68	rvmha-log-sec-ci

Table 7.1.1: rsyslog Server Interfaces

Append the following to the `/etc/hosts` file on the clustered nodes.

`/etc/hosts`:

```
10.16.136.74  rvmha-log-virt
10.16.136.67  rvmha-log-pri
192.168.0.67  rvmha-log-pri-ci
10.16.136.68  rvmha-log-sec
192.168.0.68  rvmha-log-sec-ci
```

7.2 Dynamic IP Address

The dynamic IP address used by the clients to access the rsyslog service is 10.16.136.75.

7.3 Primary Node

7.3.1 The rsyslog Daemon

The rsyslog service can receive logs from three different types of network connections: UDP, TCP, and RELP. UDP connections are used by most system logging services, but it is subject to message loss on busy networks. TCP connections do not suffer from the same message loss as UDP connections do. However, there are still times when message loss can occur. A solution to this is the RELP protocol, it helps ensure delivery of all messages.

The rsyslog servers are configured to accept all three types of connections. UDP and TCP are included in the rsyslog packages, but the RELP protocol is not. It is supplied by the **rsyslog-relp** package.



The function of inserting log entries into a PostgreSQL database is not included in the **rsyslog** package either, it is provided by the **rsyslog-pgsql** package. The **postgresql** package is not needed for the rsyslog service to write to the database, but its installation can assist in troubleshooting. Having the **postgresql** package installed allows queries to be made to the database from the rvmha-log-pri server.

Use **yum** to install the **rsyslog-pgsql**, **rsyslog-relp**, and **postgresql** packages.

```
# yum -y install rsyslog-pgsql rsyslog-relp postgresql
Loaded plugins: product-id, rhnplugin, security, subscription-manager
This system is not registered to Red Hat Subscription Management. You can
use subscription-manager to register.
This system is receiving updates from RHN Classic or RHN Satellite.
Setting up Install Process
Resolving Dependencies

[ ... Output Abbreviated ... ]

Installed:
  postgresql.x86_64 0:8.4.13-1.el6_3      rsyslog-pgsql.x86_64 0:5.8.10-6.el6
  rsyslog-relp.x86_64 0:5.8.10-6.el6

Dependency Installed:
  librelp.x86_64 0:0.1.1-4.1.el6      postgresql-libs.x86_64 0:8.4.13-1.el6_3

Complete!
```

7.3.1.1 Enable Logging to the Database

The **rsyslog-pgsql** package supplies the ability to write logs to the PostgreSQL database. The package also provides a script to aid in the proper creation of the database. As of the writing of this reference architecture, the package contains a bug that prevents the script and a supplied template from functioning correctly⁸.

The issue arises from the use of upper and lower case names used for the table and column names in the database. This bug is currently being addressed. To work around this issue, a new template is created and used in the logging rules. The creation of this template is not within the scope of this reference architecture. For further information please see the documentation for the rsyslog daemon and the PostgreSQL database.

A template called **PSQL** is created and used to log to the database:

```
$template PSQL,"INSERT INTO \"SystemEvents\"
(\"Message\", \"Facility\", \"FromHost\", \"Priority\", \"DeviceReportedTime
\", \"ReceivedAt\", \"InfoUnitID\", \"SysLogTag\") values ('%msg%',
%syslogfacility%, '%HOSTNAME%', %syslogpriority%, '%timereported:::date-
pgsql%', '%timegenerated:::date-pgsql%', %iut%, '%syslogtag%')",STDSQL
```

⁸ https://bugzilla.redhat.com/show_bug.cgi?id=928511



The `rsyslog` daemon does not have native support for logging to a PostgreSQL database, the `OMPGSQL` module provides this support. This module is loaded using the `$MODLOAD` configuration directive.

It is not desirable to send all log entries generated by remote hosts to the database and not log them to the local files on the cluster node receiving them. However, the log entries created by the local host should be written to the local files as well as the database. This can be accomplished with two rule entries in the configuration file.

The first entry uses a decision mechanism built into the `rsyslog` configuration. The `if` statement is used to detect if a condition exists and to then act upon that condition. Any logs from remote hosts that are on the local network of 10.0.0.0/8 are logged to the database. This is done with the following configuration line:

```
if $fromhost-ip startswith '10.' then :ompgsql:vmha-db-  
virt,Syslog,syslogger,[Password];PSQL
```

This line checks if the host sending the log, `$FROMHOST-IP`, starts with a 10. If it does, then log to the database using the `OMPGSQL` module. The format of the action is as follows:

```
:module:host, database, dbuser, password;template
```

The configuration line used indicates the `OMPGSQL` module is used to connect to the **vmha-db-virt** host and log entries to the **Syslog** database after authenticating using the **dbuser** and **password** credentials. Entries to the database are to be done using the **PSQL** template.

After a log entry received from a remote host is written to the database, no entries should be written to the local log files. Normally the `rsyslog` daemon continues to process log rules, this behavior is changed on a per rule basis by following the rule with a line containing “`$ ~`”. For example:

```
if $fromhost-ip startswith '10.' then :ompgsql:vmha-db-  
virt,Syslog,syslogger,[Password];PSQL  
& ~
```

The first rule ensures that all logs from remote hosts are written to the database, but it does not log entries created by the local host to the database. To do this, a simple rule that matches all logs the `rsyslog` daemon receives is created to write the entries to the database. This rule is:

```
*.* :ompgsql:vmha-db-virt,Syslog,syslogger,[Password];PSQL
```

It is very important that this rule appears after the first rule created. Otherwise log entries from the remote servers would be written to the database twice.



Create a file in the `/etc/rsyslog.d` directory called `log2db.conf`. Place the configuration inside this file.

`/etc/rsyslog.d/log2db.conf`:

```
# Allows logging to PostgreSQL database.
$ModLoad ompgsql

$template PSQL,"INSERT INTO \"SystemEvents\"
(\"Message\", \"Facility\", \"FromHost\", \"Priority\", \"DeviceReportedTime
\", \"ReceivedAt\", \"InfoUnitID\", \"SysLogTag\") values ('%msg%',
%syslogfacility%, '%HOSTNAME%', %syslogpriority%, '%timereported:::date-
pgsql%', '%timegenerated:::date-pgsql%', %iut%, '%syslogtag%')",STDSQL

#### RULES ####

if $fromhost-ip startswith '10.' then :ompgsql:rvmha-db-
virt,Syslog,syslogger,[Password];PSQL
& ~

*. * :ompgsql:rvmha-db-virt,Syslog,syslogger,[Password];PSQL
```

7.3.1.2 Allow Remote Logs to be Received

To enable `rsyslog` to receive logs from remote hosts, the correct module must be loaded and a listener must be started on the correct port.

The modules used to receive logs from the network are `IMUDP`, `IMTCP`, and `IMRELP`. The names of these modules reflect the protocol enabled by each. The modules are loaded using the `$MODLOAD` directive.

After an input module is loaded, a listener must be enabled to listen on a port. This is done using the `$UDPSERVERRUN`, `$INPUTTCPSERVERRUN`, and `$INPUTRELPSERVERRUN` directives. These directives are only available after the correct input module is loaded.

By default, system logging services send remote logs to port 514 using the UDP protocol. Enable the UDP listener to use port 514. The TCP service and RELP service do not have a standard port defined, arbitrary ports of 5514 and 6514 are used for them.



Create a file under `/etc/rsyslog.d` called `allow_remote.conf` and place the following configuration within it.

`/etc/rsyslog.d/allow_remote.conf`:

```
# Provides UDP syslog reception
$ModLoad imudp
$UDPServerRun 514

# Provides TCP syslog reception
$ModLoad imtcp
$InputTCPServerRun 5514

# Provides RELP syslog reception
$ModLoad imrelp
$InputRELPServerRun 6514
```

7.3.1.3 Restart the Service

The rsyslog service should be enabled by default, but the service needs to be restarted to allow the changes to take effect.

```
# service rsyslog restart
Shutting down system logger: [ OK ]
Starting system logger: [ OK ]
```

If an error occurs, view `/var/log/messages` for indications of the issue.

7.3.1.4 Firewall Configuration

Open the appropriate ports for the rsyslog service and save the configuration.

```
# iptables -I INPUT --match multiport --protocol tcp --dports 5514,6514 -j
ACCEPT

# iptables -I INPUT --match udp --protocol udp --dport 514 -j ACCEPT

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



7.3.2 SELinux

SELinux policies only permit the `rsyslog` daemon to use UDP port 514, UDP port 6514, and TCP port 6514. This means SELinux prevents the current `rsyslog` configuration from receiving log entries using its TCP module since that module is configured to use TCP port 5514.

SELinux can easily be configured to allow the `rsyslog` daemon to also use TCP 5514. The **`semanage`** command is needed to make the configuration change and it is provided in the **`policycoreutils-python`** package.

Install the package.

```
# yum -y install policycoreutils-python
Loaded plugins: product-id, rhnplugin, security, subscription-manager
This system is not registered to Red Hat Subscription Management. You can
use subscription-manager to register.
This system is receiving updates from RHN Classic or RHN Satellite.
```

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

```
Installed:
  policycoreutils-python.x86_64 0:2.0.83-19.30.el6
```

Complete!

Once the package is installed, the **`semanage`** command is used to list the ports that the `rsyslog` service is currently allowed to access.

```
# semanage port --list | grep syslogd_port_t
syslogd_port_t      tcp      6514
syslogd_port_t      udp      514, 6514
```

To allow `rsyslog` to access TCP port 5514, execute the following command.

```
# semanage port --add --type syslogd_port_t --proto tcp 5514
```

The port is now defined so it can be used by the `rsyslog` daemon.

```
# semanage port --list | grep syslogd_port_t
syslogd_port_t      tcp      5514, 6514
syslogd_port_t      udp      514, 6514
```



7.3.3 Database Creation

A database must be created for the log entries to be written to. The **rsyslog-pgsql** package provides the file `/usr/share/doc/rsyslog-pgsql-5.8.10/createDB.sql`. This file provides an easy way to create the database. However, as mentioned earlier, there is currently a bug that prevents this file from being used in its current state. A working version of this file is provided below.

Create a file under the `/root` directory that contains the working `createDB.sql` file as listed below.

Working `createDB.sql`:

```
CREATE DATABASE "Syslog" WITH ENCODING 'SQL_ASCII' TEMPLATE template0;
\c Syslog;
CREATE TABLE "SystemEvents"
(
    "ID" serial not null primary key,
    "CustomerID" bigint,
    "ReceivedAt" timestamp without time zone NULL,
    "DeviceReportedTime" timestamp without time zone NULL,
    "Facility" smallint NULL,
    "Priority" smallint NULL,
    "FromHost" varchar(60) NULL,
    "Message" text,
    "NTSeverity" int NULL,
    "Importance" int NULL,
    "EventSource" varchar(60),
    "EventUser" varchar(60) NULL,
    "EventCategory" int NULL,
    "EventID" int NULL,
    "EventBinaryData" text NULL,
    "MaxAvailable" int NULL,
    "CurrUsage" int NULL,
    "MinUsage" int NULL,
    "MaxUsage" int NULL,
    "InfoUnitID" int NULL ,
    "SysLogTag" varchar(60),
    "EventLogType" varchar(60),
    "GenericFileName" VarChar(60),
    "SystemID" int NULL
);

CREATE TABLE "SystemEventsProperties"
(
    "ID" serial not null primary key,
    "SystemEventID" int NULL ,
    "ParamName" varchar(255) NULL ,
    "ParamValue" text NULL
);
```



This script must be run on the database server as the *postgres* user. Use **ssh** to execute the commands in the script on the database server. By default, the *postgres* user is not allowed to login to the system using **ssh**. Login to the system using the root account instead.

```
# cat /root/createDB.sql | ssh root@rvmha-db-virt "psql -U postgres"
```

```
Warning: Permanently added 'rvmha-db-virt,10.16.136.75' (RSA) to the list of
known hosts.
root@rvmha-db-virt's password: [Password]
```

```
CREATE DATABASE
```

```
You are now connected to database "Syslog".
```

```
NOTICE: CREATE TABLE will create implicit sequence "SystemEvents_ID_seq"
for serial column "SystemEvents.ID"
```

```
NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index
"SystemEvents_pkey" for table "SystemEvents"
```

```
CREATE TABLE
```

```
NOTICE: CREATE TABLE will create implicit sequence
"SystemEventsProperties_ID_seq" for serial column
"SystemEventsProperties.ID"
```

```
NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index
"SystemEventsProperties_pkey" for table "SystemEventsProperties"
```

```
CREATE TABLE
```

The database and tables are now created. A database user must be created on the database server. The user must be given permission to access the tables and write entries into them. This user should not be given the ability to delete entries from the database as this could allow an attacker to remove log entries and any trace of malicious conduct.

The creation of the user must be performed as the *postgres* user. Since the *postgres* user does not have permission to access the database remotely, the following steps must be performed on the database server.

The **ssh** command is used to access the database server.

```
# ssh root@rvmha-db-virt
```

```
root@rvmha-db-virt's password: [Password]
```

```
Last login: Mon Apr 1 14:43:15 2013 from ovpn-113-73.phx2.redhat.com
```

```
Kickstarted on 2013-04-01
```

Using the **psql** command, create the user *syslogger* and give the user a password.

```
rvmha-db-pri# psql -U postgres -c "CREATE USER syslogger WITH PASSWORD
'[Password]';"
CREATE ROLE
```



The newly created user must have SELECT and INSERT privileges on the tables in the database. The user must also have the USAGE permission on the sequences used within the database tables.

```
rvmha-db-pri# psql -U postgres -d Syslog -c 'GRANT SELECT, INSERT ON
"SystemEvents", "SystemEventsProperties" TO syslogger;'
GRANT

rvmha-db-pri# psql -U postgres -d Syslog -c 'GRANT USAGE ON SEQUENCE
"SystemEvents_ID_seq", "SystemEventsProperties_ID_seq" TO syslogger;'
GRANT
```

The *syslogger* user must also be given permission to access the database using remote connections. This is done by making an entry in the */var/share_db/data/pg_hba.conf* file.

```
rvmha-db-pri# echo "host Syslog syslogger 10.16.136.0/21 md5" >>
/var/share_db/data/pg_hba.conf
```

The ***postgres*** cluster service must be restarted to cause it to re-read the *pg_hba.conf* file.

```
rvmha-db-pri# clusvcadm -R postgres
Local machine trying to restart service:postgres...Success
```




7.3.4 Ensure rsyslog Logs to the Database

The rsyslog daemon should be writing logs to the database. To test this, restart the rsyslog service. This should cause a stop and a start entry to be logged to the PostgreSQL database.

```
# service rsyslog restart
Shutting down system logger: [ OK ]
Starting system logger: [ OK ]
```

Use the **psql** command to query all the entries from the **SystemEvents** table of the **Syslog** database. Entries should exist for the service starts and stops.

```
# psql -U sysloger -h rvmha-db-virt -d Syslog -c 'SELECT * FROM
"SystemEvents";'
Password for user sysloger: [Password]

 ID | CustomerID | ReceivedAt | DeviceReportedTime | Facility |
Priority | FromHost |
Message | NTSeverity |
Importance | EventSource | EventUser | EventCategory | EventID |
EventBinaryData | MaxAvailable | CurrUsage | MinUsage | MaxUsage |
InfoUnitID | SysLogTag | EventLogType | GenericFileName | SystemID
-----+-----+-----+-----+-----+
+-----+-----+
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
7 | | 2013-04-11 14:23:53 | 2013-04-11 14:23:53 | 0 |
6 | rvmha-log-pri | Kernel logging (proc) stopped.
| | | | | |
1 | kernel: | | | | |
8 | | 2013-04-11 14:23:53 | 2013-04-11 14:23:53 | 5 |
6 | rvmha-log-pri | [origin software="rsyslogd" swVersion="5.8.10" x-
pid="4399" x-info="http://www.rsyslog.com"] exiting on signal 15. |
| | | | | 1 | rsyslogd: |
9 | | 2013-04-11 14:23:53 | 2013-04-11 14:23:53 | 0 |
6 | rvmha-log-pri | imklog 5.8.10, log source = /proc/kmsg started.
| | | | | |
1 | kernel: | | | | |
10 | | 2013-04-11 14:23:53 | 2013-04-11 14:23:53 | 5 |
6 | rvmha-log-pri | [origin software="rsyslogd" swVersion="5.8.10" x-
pid="4423" x-info="http://www.rsyslog.com"] start
| | | | | 1 | rsyslogd: |
(4 rows)
```



7.3.5 Cluster Configuration

At this point, the `rvmha-log-pri` primary node should be configured and it should be writing log entries to the PostgreSQL database. The next step is to configure the cluster software.

Configuring the primary cluster node is done using steps similar to that of the PostgreSQL server. Since the steps were explained during the creation of the database service creation, they are not explained along with the steps.

The clustered nodes have a network interface on a cluster network. The interface names are **`rvmha-log-pri-ci`** and **`rvmha-log-sec-ci`**. A virtual IP address of 10.16.136.74 is used for the cluster service and has a resolvable name of **`rvmha-log-virt`**.

7.3.5.1 Define a Cluster and Add Nodes

Define a cluster named *`rvmha-rsyslog`*.

```
# ccs --host rvmha-log-pri-ci --createcluster rvmha-rsyslog
rvmha-log-pri-ci password: [Password]
```

Configure the cluster as a two-node cluster.

```
# ccs -h rvmha-log-pri-ci --setcman two_node=1 expected_votes=1
```

Add the primary and secondary nodes to the cluster.

```
# ccs --host rvmha-log-pri-ci --addnode rvmha-log-pri-ci
Node rvmha-log-pri-ci added.

# ccs --host rvmha-log-pri-ci --addnode rvmha-log-sec-ci
Node rvmha-log-sec-ci added.
```



7.3.5.2 Configure Fencing

Create the fencing device.

```
# ccs --host rvmha-log-pri-ci --addfencedev RHEVM-API agent=fence_rhevm  
ipaddr=jh-rhevm.cloud.lab.eng.bos.redhat.com login="[User]@internal"  
passwd="[Password]" ssl=1
```

Add a fencing method to each node.

```
# ccs --host rvmha-log-pri-ci --addmethod API rvmha-log-pri-ci  
Method API added to rvmha-log-pri-ci.  
  
# ccs --host rvmha-log-pri-ci --addmethod API rvmha-log-sec-ci  
Method API added to rvmha-log-sec-ci.
```

Configure a fencing instance for each node.

```
# ccs --host rvmha-log-pri-ci --addfenceinst RHEVM-API rvmha-log-pri-ci API  
port=rvmha-log-pri  
  
# ccs --host rvmha-log-pri-ci --addfenceinst RHEVM-API rvmha-log-sec-ci API  
port=rvmha-log-sec
```

7.3.5.3 Configure a Failover Domain

Create a failover domain called *rsyslog_dom*.

```
# ccs --host rvmha-log-pri-ci --addfailoverdomain rsyslog_dom unrestricted  
ordered failback
```

Add the primary node to the failover domain.

```
# ccs --host rvmha-log-pri-ci --addfailoverdomainnode rsyslog_dom rvmha-log-  
pri-ci 1
```

7.3.5.4 Define rsyslog Service

```
# ccs --host rvmha-log-pri-ci --addservice rsyslog domain=rsyslog_dom  
autostart=1 recovery=relocate
```

```
# ccs --host rvmha-log-pri-ci --addsubservice rsyslog ip  
address=10.16.136.74
```

```
# ccs --host rvmha-log-pri-ci --lsservices  
service: name=rsyslog, domain=rsyslog_dom, autostart=1, recovery=relocate  
ip: address=10.16.136.74  
resources:
```



7.4 Secondary Node

7.4.1 The rsyslog Daemon

Use **yum** to install the **rsyslog-pgsql**, **rsyslog-relp**, and **postgresql** packages.

```
# yum -y install rsyslog-pgsql rsyslog-relp postgresql
Loaded plugins: product-id, rhnplugin, security, subscription-manager
This system is not registered to Red Hat Subscription Management. You can
use subscription-manager to register.
This system is receiving updates from RHN Classic or RHN Satellite.
Setting up Install Process
Resolving Dependencies

[ ... Output Abbreviated ... ]

Installed:
  postgresql.x86_64 0:8.4.13-1.el6_3      rsyslog-pgsql.x86_64 0:5.8.10-6.el6
  rsyslog-relp.x86_64 0:5.8.10-6.el6

Dependency Installed:
  librelp.x86_64 0:0.1.1-4.1.el6      postgresql-libs.x86_64 0:8.4.13-1.el6_3

Complete!
```

7.4.1.1 Enable Logging to the Database

Since the configuration file that allows logging to the database exists on the primary node, copy it from there.

```
# scp rvmha-log-pri:/etc/rsyslog.d/log2db.conf /etc/rsyslog.d
root@rvmha-log-pri's password: [Password]
log2db.conf                                100% 572      0.6KB/s   00:00
```

7.4.1.2 Allow Remote Logs to be Received

Copy the configuration file that allows remote connections.

```
# scp rvmha-log-pri:/etc/rsyslog.d/allow_remote.conf /etc/rsyslog.d
root@rvmha-log-pri's password: [Password]
allow_remote.conf                          100% 220      0.2KB/s   00:00
```

7.4.1.3 Restart the Service

Restart the **rsyslog** service so it re-reads the configuration files.

```
# service rsyslog restart
Shutting down system logger:                [ OK ]
Starting system logger:                      [ OK ]
```



7.4.1.4 Firewall Configuration

Open the appropriate ports for the `rsyslog` service and save the configuration.

```
# iptables -I INPUT --match multiport --protocol tcp --dports 5514,6514 -j ACCEPT

# iptables -I INPUT --match udp --protocol udp --dport 514 -j ACCEPT

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

7.4.2 SELinux

Install the **polycoreutils-python** package.

```
# yum -y install polycoreutils-python
Loaded plugins: product-id, rhnplugin, security, subscription-manager
This system is not registered to Red Hat Subscription Management. You can
use subscription-manager to register.
This system is receiving updates from RHN Classic or RHN Satellite.

[ ... Output Abbreviated ... ]

Installed:
  polycoreutils-python.x86_64 0:2.0.83-19.30.el6

Complete!
```

Use the **semanage** command to allow `rsyslog` to access TCP port 5514.

```
# semanage port --add --type syslogd_port_t --proto tcp 5514
```

Make sure the port is defined.

```
# semanage port --list | grep syslogd_port_t
syslogd_port_t          tcp      5514, 6514
syslogd_port_t          udp      514, 6514
```



7.5 Starting the Cluster

On the primary node, issue the **ccs_sync** command to propagate the *cluster.conf* file to the secondary node.

```
# ccs_sync
You have not authenticated to the ricci daemon on rvmha-log-pri-ci
Password: [Password]
You have not authenticated to the ricci daemon on rvmha-log-sec-ci
Password: [Password]
```

Use the **ccs** command to start the cluster services on all the cluster nodes.

```
# ccs --host rvmha-log-pri-ci --startall
rvmha-db-sec-ci password: [Password]
Started rvmha-log-sec-ci
Started rvmha-log-pri-ci
```

Use the **clustat** command to see the status of the cluster and its services.

```
# clustat
Cluster Status for rvmha-rsyslog @ Thu Apr 11 14:37:17 2013
Member Status: Quorate

Member Name                ID    Status
-----
rvmha-log-pri-ci           1 Online, Local, rgmanager
rvmha-log-sec-ci           2 Online, rgmanager

Service Name                Owner (Last)                State
-----
service:rsyslog             rvmha-log-pri-ci            started
```

Now that cluster service is up and functioning, test to make sure the service can stat on both nodes.

```
# clusvcadm -r rsyslog -m rvmha-log-sec-ci
Trying to relocate service:rsyslog to rvmha-log-sec-ci...Success
service:rsyslog is now running on rvmha-log-sec-ci

# clustat
Cluster Status for rvmha-rsyslog @ Thu Apr 11 14:39:30 2013
Member Status: Quorate

Member Name                ID    Status
-----
rvmha-log-pri-ci           1 Online, Local, rgmanager
rvmha-log-sec-ci           2 Online, rgmanager

Service Name                Owner (Last)                State
-----
service:rsyslog             rvmha-log-sec-ci            started
```



Make sure the service can be restarted on the primary node.

```
# clusvcadm -r rsyslog -m rvmha-log-pri-ci
Trying to relocate service:rsyslog to rvmha-log-pri-ci...Success
service:rsyslog is now running on rvmha-log-pri-ci

# clustat
Cluster Status for rvmha-rsyslog @ Thu Apr 11 20:51:33 2013
Member Status: Quorate

Member Name          ID    Status
-----
rvmha-log-pri-ci      1    Online, Local, rgmanager
rvmha-log-sec-ci      2    Online, rgmanager

Service Name          Owner (Last)          State
-----
service:rsyslog       rvmha-log-pri-ci      started
```

7.6 Clients

This section shows how to configure three logging clients to send all logs generated on them to the logging server. Since the logging server is configured to write the logs to the PostgreSQL database, the logs can be viewed with calls to the database.

7.6.1 UDP Client

Configuring the **rvmha-srv-udp** system to send its log messages to the logging server is done by creating a single rule in the configuration file.

The rule is in the format of:

```
service.priority @hostname:port;template
```

To match all services and priorities, use an *ASTERISK* for each. The single @ symbol is used to indicate the system should use the UDP protocol when sending the logs messages to the remote server. The template *RSYSLOG_FORWARDFORMAT* is a standard format that can be used. See the **rsyslog** documentation for more information on other available templates.

Create a file called *udp2remote.conf* in the */etc/rsyslog.d* directory that contains the rule.

/etc/rsyslog.d/udp2remote.conf:

```
#### RULES ####
*.* @rvmha-log-virt:514;RSYSLOG_ForwardFormat
```



Once the `rsyslog` service is restarted, the client should start logging messages to the remote server.

```
# service rsyslog restart
Shutting down system logger:      [ OK ]
Starting system logger:           [ OK ]
```

On the `rvmha-db-pri` system, use the `psql` command to check if logs are written to the database for the `rvmha-srv-udp` server.

```
# psql -U postgres -d Syslog
psql (8.4.13)
Type "help" for help.

Syslog=# select "DeviceReportedTime", "FromHost", "Message" from
"SystemEvents" where "FromHost"='rvmha-srv-udp';

 DeviceReportedTime | FromHost |
Message
-----+-----
+-----+-----
2013-04-11 16:10:55 | rvmha-srv-udp | Kernel logging (proc) stopped.
2013-04-11 16:10:55 | rvmha-srv-udp | [origin software="rsyslogd"
swVersion="5.8.10" x-pid="2607" x-info="http://www.rsyslog.com"] exiting on
signal 15.
2013-04-11 16:10:55 | rvmha-srv-udp | imklog 5.8.10, log source =
/proc/kmsg started.
2013-04-11 16:10:55 | rvmha-srv-udp | [origin software="rsyslogd"
swVersion="5.8.10" x-pid="2628" x-info="http://www.rsyslog.com"] start
(4 rows)
```

7.6.2 TCP Client

Configuring the `rvmha-srv-tcp` client to send log messages using TCP requires a simple rule as well. The rule has the same format as sending UDP with one exception, two `@` symbols are used instead of one.

Create a file called `tcp2remote.conf` in the `/etc/rsyslog.d` directory that contains the rule.

`/etc/rsyslog.d/tcp2remote.conf`:

```
#### RULES ####
*. * @@rvmha-log-virt:5514;RSYSLOG_ForwardFormat
```




Because the `rsyslog` daemon does not have access to TCP port 5514, an entry must be made to the SELinux configuration to allow the access.

Install the **polycoreutils-python** package.

```
# yum -y install polycoreutils-python
Loaded plugins: product-id, rhnplugin, security, subscription-manager
This system is not registered to Red Hat Subscription Management. You can
use subscription-manager to register.

[ ... Output Abbreviated ... ]

Installed:
  polycoreutils-python.x86_64 0:2.0.83-19.30.el6

Complete!
```

Use the **semanage** command to allow the `rsyslog` service to access TCP port 5514.

```
# semanage port -a -t syslogd_port_t -p tcp 5514
```

Make sure the port is defined for use by `rsyslog`.

```
# semanage port --list | grep syslogd_port_t
syslogd_port_t          tcp      5514, 6514
syslogd_port_t          udp      514, 6514
```

Once the `rsyslog` service is restarted, the client should start logging messages to the remote server.

```
# service rsyslog restart
Shutting down system logger:          [ OK ]
Starting system logger:                [ OK ]
```

On the `rvmha-db-pri` system, use the **psql** command to check if logs are written to the database for the `rvmha-srv-tcp` server.

```
# psql -U postgres -d Syslog
psql (8.4.13)
Type "help" for help.

Syslog=# select "DeviceReportedTime", "FromHost", "Message" from
"SystemEvents" where "FromHost"='rvmha-srv-tcp';
 DeviceReportedTime | FromHost |
Message
-----+-----
+-----
2013-04-11 16:20:26 | rvmha-srv-tcp | imklog 5.8.10, log source =
/proc/kmsg started.
2013-04-11 16:20:26 | rvmha-srv-tcp | [origin software="rsyslogd"
swVersion="5.8.10" x-pid="2623" x-info="http://www.rsyslog.com"] start
(2 rows)
```



7.6.3 RELP Client

Configuring the **rvmha-srv-relp** client to send log messages using the RELP protocol requires that the **OMRELP** module is loaded in the **rsyslog** configuration. This module is provided by the **rsyslog-relp** package. Install the package.

```
# yum -y install rsyslog-relp
Loaded plugins: product-id, rhnplugin, security, subscription-manager
This system is not registered to Red Hat Subscription Management. You can
use subscription-manager to register.
This system is receiving updates from RHN Classic or RHN Satellite.
Setting up Install Process
Resolving Dependencies
```

[... Output Abbreviated ...]

```
Installed:
  rsyslog-relp.x86_64 0:5.8.10-6.el6
```

```
Dependency Installed:
  librelp.x86_64 0:0.1.1-4.1.el6
```

```
Complete!
```

Once the package is installed, the module is loaded using the **\$MODLOAD** directive in the configuration file. Once the module is loaded, a rule is created to send the logs to the remote server using the protocol.

Instead of using **AT** symbols in the rule definition, the module name enclosed in colons is specified instead.

Create a file called **relp2remote.conf** in the **/etc/rsyslog.d** directory that contains the rule.

/etc/rsyslog.d/relp2remote.conf:

```
# Provides RELP syslog reception
$ModLoad omrelp

#### RULES ####
*. * :omrelp:rvmha-log-virt:6514;RSYSLOG_ForwardFormat
```

Once the **rsyslog** service is restarted, the client should start logging messages to the remote server.

```
# service rsyslog restart
Shutting down system logger: [ OK ]
Starting system logger: [ OK ]
```



On the rvmha-db-pri system, use the **psql** command to check if logs are written to the database for the rvmha-srv-tcp server.

```
# psql -U postgres -d Syslog
psql (8.4.13)
Type "help" for help.

Syslog=# select "DeviceReportedTime", "FromHost", "Message" from
"SystemEvents" where "FromHost"='rvmha-srv-relp';
 DeviceReportedTime | FromHost |
Message
-----+-----
2013-04-11 16:26:48 | rvmha-srv-relp | imklog 5.8.10, log source =
/proc/kmsg started.
2013-04-11 16:26:48 | rvmha-srv-relp | [origin software="rsyslogd"
swVersion="5.8.10" x-pid="2704" x-info="http://www.rsyslog.com"] start
(2 rows)
```



8 Testing Fail Over

To test that the environment works as expected, power management on the hypervisors must be disabled. In the Red Hat Enterprise Virtualization Manager, edit the hypervisors properties under the **Hosts** tab. Once the **Edit Host** window appears, select **Power Management** and uncheck the **Enable Power Management** check box. Do this for both hypervisors.

Power off hypervisor rvmha-rhevh1 without shutting the hypervisor down. This simulates a power failure on the hypervisor.

The **clustat** command on the rvmha-db-sec cluster node indicates the rvmha-db-pri-ci cluster member is offline, but that the *POSTGRES* service is still running on rvmha-db-pri-ci server. This happens because rvmha-rhevh1 was the Storage Pool Manager (SPM) of the cluster.

Check the cluster status on rvmha-db-sec.

```
# clustat
Cluster Status for rvmha-db @ Fri Apr 12 00:10:51 2013
Member Status: Quorate

Member Name          ID    Status
-----
rvmha-db-pri-ci      1 Offline
rvmha-db-sec-ci      2 Online, Local, rgmanager

Service Name          Owner (Last)          State
-----
service:postgres      rvmha-db-pri-ci      started
```

The service remains in this state until a new SPM is elected. Once a new storage pool manager is elected, the cluster software can continue to fail the service over to the secondary node.



Once the service is up on the rvmha-db-sec-ci node, test the service by restarting the rsyslog service on the rvmha-srv-relp client. This causes a log to be written to the logging server. The log is seen by querying the Syslog database.

```
Syslog=# select "DeviceReportedTime", "FromHost", "Message" from
"SystemEvents" where "FromHost"='rvmha-srv-relp';

DeviceReportedTime | FromHost |
Message
-----+-----
+-----+-----
2013-04-12 00:42:51 | rvmha-srv-relp | Kernel logging (proc) stopped.
2013-04-12 00:42:51 | rvmha-srv-relp | [origin software="rsyslogd"
swVersion="5.8.10" x-pid="1731" x-info="http://www.rsyslog.com"] exiting on
signal 15.
2013-04-12 00:42:52 | rvmha-srv-relp | imklog 5.8.10, log source =
/proc/kmsg started.
2013-04-12 00:42:52 | rvmha-srv-relp | [origin software="rsyslogd"
swVersion="5.8.10" x-pid="1836" x-info="http://www.rsyslog.com"] start
(4 rows)
```

Power the rvmha-rhevh1 hypervisor on. Make sure the virtual machines start. Start and virtual machines that do not start automatically.

The rvmha-db-pri server fences the rvmha-db-sec when it boots. This is normal and ensures that the rvmha-db-sec server is not accessing any resources when the cluster relocates the service back to the rvmha-db-pri server.

Power off hypervisor rvmha-rhevh2 without shutting the hypervisor down.

The **clustat** command on the rvmha-log-sec cluster node indicates the rvmha-log-pri-ci cluster member is offline, but that the *RSYSLOG* service is still running on rvmha-log-pri-ci server. Again, this is due to Storage Pool Elections.

Check the cluster status on rvmha-log-sec.

```
# clustat
Cluster Status for rvmha-rsyslog @ Fri Apr 12 01:17:51 2013
Member Status: Quorate

Member Name          ID   Status
-----
rvmha-log-pri-ci      1   Offline
rvmha-log-sec-ci      2   Online, Local, rgmanager

Service Name          Owner (Last)          State
-----
service:rsyslog       rvmha-log-pri-ci      started
```



Once the service is up on the rvmha-log-sec-ci node, test the service by restarting the rsyslog service on the rvmha-srv-tcp client. This causes a log to be written to the logging server. The log is seen by querying the Syslog database.

```
Syslog=# select "DeviceReportedTime", "FromHost", "Message" from
"SystemEvents" where "FromHost"='rvmha-srv-tcp';
 DeviceReportedTime | FromHost |
Message
-----+-----
+-----
2013-04-12 01:48:54 | rvmha-srv-tcp | imklog 5.8.10, log source =
/proc/kmsg started.
2013-04-12 01:48:54 | rvmha-srv-tcp | [origin software="rsyslogd"
swVersion="5.8.10" x-pid="1853" x-info="http://www.rsyslog.com"] start
(2 rows)
```



9 Conclusion

This reference architecture demonstrated the use of the Red Hat Enterprise Linux High Availability Add-On running in virtual machines running within the Red Hat Enterprise Virtualization solution.

Two separate two node clusters were created, each with a primary and secondary node. These two-node clusters were configured to work in an active-passive manner. Each primary node ran on separate hypervisors. The secondary nodes ran on a different hypervisor from their primary node counterpart. This was achieved by “pinning” the virtual machines to a particular hypervisor.

One cluster contained a virtual IP address for the rsyslog server to receive event logs on. Once the event was received the rsyslog daemon wrote the event to a PostgreSQL database. This PostgreSQL database instance was clustered on the second cluster, using a shared iSCSI LUN to store the database files.

Clustering fail over was tested by essentially pulling the power from each hypervisor and checking the PostgreSQL database to see if new log entries were being made.

The goal of demonstrating the use of the Red Hat Enterprise Linux High Availability Add-On inside virtual machines within the Red Hat Enterprise Virtualization environment were achieved, showing that a viable clustered solution is available for the Red Hat Enterprise Virtualization solution.



Appendix A: Configuration Files

The configuration files used during the creation of this reference architecture can be downloaded⁹ from the Red Hat Customer Portal in a compressed tar archive. A description of the files contained within the archive are listed in **Table 9.1: Configuration Files**.

Virtual Machine	Tar Archive Filename	Node Filename
rvmha-db-{pri,sec}	db-multipath.conf	/etc/multipath.conf
	db-cluster.conf	/etc/cluster/cluster.conf
	mypol.te	/root/mypol.te
rvmha-log-pri	createDB.sql	/root/createDB.sql
rvmha-log-{pri,sec}	log-cluster.conf	/etc/cluster/cluster.conf
	allow_remote.conf	/etc/rsyslog.d/allow_remote.conf
	log2db.conf	/etc/rsyslog.d/log2db.conf
rvmha-srv-udp	udp2remote.conf	/etc/rsyslog.d/udp2remote.conf
rvmha-srv-tcp	tcp2remote.conf	/etc/rsyslog.d/tcp2remote.conf
rvmha-srv-relp	relp2remote.conf	/etc/rsyslog.d/relp2remote.conf

Table 9.1: Configuration Files

⁹ <https://access.redhat.com/site/node/353073/40/1>



Appendix B: Channel Subscriptions

This reference architecture uses Classic Red Hat Network channel subscriptions throughout it. The newer Red Hat Subscription Manager uses different names for the channels that contain the packages needed to complete this paper. Table **Table 9.2: Channel Maps** maps the names of the Classic RHN channel names used in this reference architecture to the names of the repositories used when using Subscription Manager.

Channel	Classic Channel	Subscription Management Repository
Red Hat Enterprise Virtualization Guest Agent	rhel-x86_64-rhev-agent-6-server	rhel-6-server-rhev-mgmt-agent-rpms
High Availability Add-On	rhel-x86_64-server-ha-6	rhel-ha-for-rhel-6-server-rpms

Table 9.2: Channel Maps



Appendix C: Resources

This Reference Architecture and Configuration Files

<https://access.redhat.com/site/articles/353073>

Red Hat Enterprise Virtualization Documentation

https://access.redhat.com/site/documentation/Red_Hat_Enterprise_Virtualization

Red Hat Enterprise Linux 6 - Cluster Administration

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

HA Resources and Parameters

https://access.redhat.com/site/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Cluster_Administration/ap-ha-resource-params-CA.html

SELinux Guide

https://access.redhat.com/site/documentation/en-US/Red_Hat_Enterprise_Linux/4/html/SELinux_Guide/selg-preface-0011.html

Virtualization Support for High Availability in Red Hat Enterprise Linux 5 and 6

<https://access.redhat.com/knowledge/articles/29440>

Red Hat Enterprise Linux Cluster, High Availability, and GFS Deployment Best Practices

<https://access.redhat.com/knowledge/articles/40051>

Red Hat Enterprise Linux 6.3 Technical Notes - scsi_dh Modules

https://access.redhat.com/knowledge/docs/en-US/Red_Hat_Enterprise_Linux/6/html-single/6.3_Technical_Notes/index.html#kernel_issues

See **kernel component**, **BZ#690523**

Linux DM Installation Details for Dell PowerVault MD Series Storage Arrays

<http://www.dell.com/downloads/global/products/pvaul/en/powervault-md36x0i-linux-dm-installation.pdf>

Bugzilla: Postgres8 Monitoring Service Fails

https://bugzilla.redhat.com/show_bug.cgi?id=814334

Bugzilla: rsyslog createDB.sql contains errors

https://bugzilla.redhat.com/show_bug.cgi?id=928511

802 IEEE 802® Standard

<http://standards.ieee.org/getieee802/download/802-2001.pdf>



Appendix D: Contributors

Scott Collier, Senior Principal Software Engineer
Paper Review

Steve Reichard, Senior Principal Software Engineer
Technical Assistance and Troubleshooting

Mark Heslin, Principal Software Engineer
Provided overview section for the **High Availability Add-On**.



Appendix E: Revision History

Revision 1.0
Initial Release

Friday April 12, 2013

John Herr

