



**Red Hat Reference Architecture Series**

# **Implementing InterSystems Caché® Database in a Red Hat Enterprise Linux High Availability Cluster**

**Don Frederick**  
Senior Healthcare EA  
RHCE, RHCVA, RHCDS

**Version 1.1**  
**January 31, 2012**





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.

Microsoft and Windows are U.S. registered trademarks of Microsoft Corporation.

UNIX is a registered trademark of The Open Group.

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

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

© 2011 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](mailto: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

Send feedback to [refarch-feedback@redhat.com](mailto:refarch-feedback@redhat.com)



## Table of Contents

1 Executive Summary.....	1
2 InterSystems Caché® Solution Overview.....	2
3 Red Hat Clustering Overview.....	3
4 Solution Requirements.....	4
4.1 Hardware Requirements.....	4
4.2 Software Requirements.....	4
5 Reference Environment Overview.....	6
5.1 Reference Environment Detail.....	6
6 Software Installation.....	8
6.1 Software Installation - Red Hat Enterprise Linux.....	8
6.2 Software Installation - Cluster Software.....	11
6.3 Software Installation - Caché Database.....	15
6.4 Software Installation - Summary.....	16
7 Software Configuration.....	17
7.1 Software Configuration - Network Bonding.....	17
7.2 Software Configuration - The Storage Network.....	21
7.3 Software Configuration - Red Hat Enterprise Linux Multipath.....	24
7.4 Software Configuration - Prepare Cluster Environment.....	25
7.5 Software Configuration - The Cluster.....	25
7.6 Software Configuration - Quorum Disk.....	29
7.7 Software Configuration - HA LVM.....	31
7.8 Software Configuration – InterSystems Caché as a Clustered Service.....	33
8 Conclusion.....	45
9 Suggested Reading and References.....	46



# 1 Executive Summary

The purpose of this document is to provide a reference for implementing the [InterSystems Caché®](#) database in a fault tolerant manner. Specifically, this guide will discuss using Red Hat Clustering for the High Availability requirement to provide the highest level of up-time and automated fail-over capability during system failures. Additionally, the guide will walk through an overview of clustering, a description of the requirements to build and implement this solution, and a description of a the common reference architecture. In addition, the document will cover the best practice for installing and configuring this reference architecture to include the operating system, clustering software and the related Caché database.



## 2 InterSystems Caché® Solution Overview

InterSystems Caché® is an advanced object database that provides in-memory speed with persistence, and the ability to handle huge volumes of transactional data. Plus, it can run SQL faster than relational databases. Caché enables rapid Web application development, massive scalability, and real-time queries against transactional data – with minimal maintenance and hardware requirements. Visit InterSystems Caché®'s [website](#) for more information about it and its software solutions.



## 3 Red Hat Clustering Overview

The Red Hat Cluster software available for all supported versions of Red Hat Enterprise Linux provides a high availability solution for software. The clustering software provides tools to deploy and manage resources and services relevant to an active cluster. In addition to the software tools that provide access and management of a cluster, the solution also includes mechanisms for detection and fencing of failed nodes of the cluster. And while the clustering provides high availability at a software level, the underlying operating system will provide networking and storage redundancy through software services such as bonding and multipathing. While these services are not formally a part of the Red Hat Clustering, both network bonding and storage multipathing are relevant for a fault tolerant solution. Here is a short listing and description of the terms, tools and components that will be discussed throughout the remainder of this document.

- **High Availability Cluster:** A managed collection of systems, devices, scripts and software that work together to meet a given SLA of a service.
- **Resource:** It is a representation of script or software or network ip that when grouped together forms a service.
- **Service:** A specific set of cluster resources that are grouped together to perform a task.
- **Failover Domain:** Represents the logical grouping of systems for which a clustered service is enabled to run.
- **Fence Device:** A mechanism that can effectively remove access of a system to clustered resources such as storage. This mechanism can be power based or storage base.
- **Quorum Disk:** This is also known as qdisk and uses a shared external storage device to keep track of a system's current status. This storage device is queried to ensure actively responsive nodes have membership to a cluster.



# 4 Solution Requirements

A Red Hat cluster configured to provide high availability to InterSystems Caché®'s database will need to consider a number of hardware and software requirements. The following two sections will discuss both the recommended hardware and software requirements.

## 4.1 Hardware Requirements

The following is a recommended list of hardware to consider prior to working through the cluster build.

- *Servers:* It is required to include at a minimum of two x86, 64 bit based server grade systems in the working cluster. The servers should include at least four network cards divided between two controllers. Each server should have at least two fibre cards and access to shared storage.
- *Fence Device:* The servers should each be attached to a fence device such as a power fence device, on board fence device or storage fence device. This [Red Hat Knowledge Base Document](#) provides a list of Red Hat supported fence devices.
- *Networks:* Each server should be connected to at least two networks with each connection being redundantly backed by two network cards. One network will be for cluster traffic, and the other network will be for application traffic. If the servers will be connected to network based storage, then a third set of network cards should be connected for storage traffic. The storage traffic should be directed over a separate isolated network for performance reasons.
- *Storage:* The servers should be connected to a SAN and atleast two luns will need to be shared to the clustered servers. The first lun will be be for storing the database and will be the largest of the luns. The second lun will be used for the quorum disk and will be roughly 10MB in size.

## 4.2 Software Requirements

Once the hardware considerations have been reviewed and met, the next set of requirements to consider will be software related. The software portion will include the operating system including the recommended core build, the clustering software and database software.

- *Operating System:* This guide is written with the intention of deploying Red Hat Enterprise Linux 5 as the base operating system on each of the clustered systems. Red Hat Enterprise Linux will be installed with a specific set of software packages to include the needed software for networking, network time keeping and storage multipathing software. Once the system is installed and configured, it is also expected that the system can always be updated to the latest service pack level since APIs and ABIs are consistent within releases. See this [document](#) for more information about the API/ABI guarantee. Here is the link to the Red Hat Enterprise Linux documentation [portal](#).
- *Cluster Software:* Red Hat Enterprise Linux includes an option for clustering software and is fully integrated in to the operating system. While there is an option for cluster storage software also known as GFS. This solution will not use GFS and only be



focusing on the high availability portions of the cluster software. Here is a link to the Red Hat Cluster Overview [document](#) for an overview on Red Hat clustering.

- *Database Software*: InterSystems Caché® will be the database that we document in this reference architecture as a clustered service. Here is the [link](#) for more information about InterSystems Caché®.



# 5 Reference Environment Overview

This reference architecture is written against a virtual environment hosted on a RHEL 6 hypervisor. The amount of memory, storage and cpus are limited, and it is recommended for production deployments that InterSystems Caché®'s hardware recommendations are reviewed. The following describes in more detail the systems and system resources that are used in this reference implementation.

## 5.1 Reference Environment Detail

- **Systems:** Two x86, 64 bit KVM Virtual Machines with identical allocated resources with the following hostnames:

```
[root@db1 ~]# hostname
db1.privnet.redhat.com
```

```
[root@db2 ~]# hostname
db2.privnet.redhat.com
```

- **Networks:** Each system was allocated 6 network cards with a pair of network cards attached to the 172.31.224.0 network and a pair attached to the 192.168.0.0 network and a pair attached to the 172.16.0.0 network.

```
### Host: db1
[root@db1 ~]# cat /etc/sysconfig/network-scripts/ifcfg-eth* | egrep -i
'(DEVICE|HWADDR)'
# Virtio Network Device
DEVICE=eth0
HWADDR=52:54:00:DB:BF:22
# Virtio Network Device
DEVICE=eth1
HWADDR=52:54:00:be:1e:94
# Virtio Network Device
DEVICE=eth2
HWADDR=52:54:00:AA:D1:A9
# Virtio Network Device
DEVICE=eth3
HWADDR=52:54:00:55:db:f1
# Virtio Network Device
DEVICE=eth4
HWADDR=52:54:00:20:c9:af
# Virtio Network Device
DEVICE=eth5
HWADDR=52:54:00:7e:2a:91

### Host: db2
[root@db2 ~]# cat /etc/sysconfig/network-scripts/ifcfg-eth* | egrep -i
'(DEVICE|HWADDR)'
# Virtio Network Device
DEVICE=eth0
HWADDR=52:54:00:CB:FA:92
# Virtio Network Device
DEVICE=eth1
```



```
HWADDR=52:54:00:3C:84:EF
# Virtio Network Device
DEVICE=eth2
HWADDR=52:54:00:46:59:47
# Virtio Network Device
DEVICE=eth3
HWADDR=52:54:00:0C:D2:01
# Virtio Network Device
DEVICE=eth4
HWADDR=52:54:00:3e:d3:28
# Virtio Network Device
DEVICE=eth5
HWADDR=52:54:00:2e:e2:fc
```

- **Storage:** Each system is allocated one 10.7 GB local disk to be used for installing Red Hat Enterprise Linux, one shared 104 MB shared disk to be used for the quorum disk and one 10.7 GB shared disk to be used for the Caché database.

```
Local Disk /dev/vda: 10.7 GB, 10737418240 bytes
```

```
Shared Disk /dev/vdb: 104 MB, 104857600 bytes
```

```
Shared Disk over multiple paths:
```

```
Disk /dev/sda: 10.7 GB, 10737418240 bytes
```

```
Disk /dev/sdb: 10.7 GB, 10737418240 bytes
```

```
[root@db1 ~]# multipath -ll
mpath0 (SATA_WDC_WD3200BEKT-_WD-WX31A70H3425) dm-1 ATA,WDC WD3200BEKT-0
[size=10G][features=0][hwhandler=0][rw]
\_ round-robin 0 [prio=2][active]
  \_ 2:0:0:1 sda 8:0 [active][ready]
  \_ 3:0:0:1 sdb 8:16 [active][ready]
```

- **Memory:** Each system has 2048 MB allocated.
- **CPUS:** Each system has two cpus allocated.



# 6 Software Installation

Now that you have worked your way through the requirements of the project, the next step in the process will be to install Red Hat Enterprise Linux 5 (latest minor release), the cluster software groups and the database software.

## 6.1 Software Installation - Red Hat Enterprise Linux

The simplest way to install the operating system will be to use a dvd and working your way through the installer. You also have the option of installing Red Hat Enterprise Linux over http, ftp or nfs. If you would like to automate the install, a recommended approach would be to use a kickstart file. The kickstart file can be edited to include any specific pre-scripts, specific environment settings, list of packages and additional post scripts for configuring the final state of the system. The kickstart file provides a consistent way to build virtual and physical systems and is considered to be a recipe based approach. Here is an example of a minimal kickstart file.

Note: In order to ensure that the installation is placed on local storage, it is advised that you remove the system from any access to shared storage. If access is not removed, it is possible that you will install the OS over the shared LUN.

```
# Kickstart file automatically generated by anaconda.

install
url --url http://172.31.224.254/pub/distro/rhel-5-server/u5/x86_64/os
key --skip
lang en_US.UTF-8
keyboard us
xconfig --startxonboot
network --device eth0 --bootproto dhcp
firewall --enabled --port=22:tcp
authconfig --enablesshadow --enablemd5
selinux --permissive
timezone --utc America/New_York
bootloader --location=mbr --driveorder=hda --append="rhgb quiet"
# The following is the partition information you requested
# Note that any partitions you deleted are not expressed
# here so unless you clear all partitions first, this is
# not guaranteed to work
#clearpart --linux --drives=sda
#part /boot --fstype ext3 --size=100 --ondisk=sda
#part pv.6 --size=0 --grow --ondisk=sda
#volgroup vgroot --pesize=32768 pv.6
#logvol / --fstype ext3 --name=vgroot --vgname=vgroot --size=1024 --grow
#logvol swap --fstype swap --name=lvroot --vgname=vgroot --size=1008 --grow
--maxsize=2016

%packages
@base
@core
@base-x
device-mapper-multipath
emacs
```



```
xorg-x11-utils
%post
rpm --import /etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release
```

- In order to use the kickstart approach, you will need to provide the first stage of the install to each system. This can be done by booting each host with a boot.iso, pxe or other chosen method. Then either enter 'linux ks=http://link/to/ks.cfg' at the boot prompt for the web hosted kickstart or choose the menu item to install from within your pxe server boot menu.

Note: One of the items to consider when configuring a cluster is to ensure that all hostnames are resolvable. It is always good practice to add each host to dns and also to the '/etc/hosts' file on each node. The hosts information could be added to each system during the initial build with kickstart.

- In the example kickstart, the partitioning information was left out, so you will be prompted to choose the install device and the partitioning scheme.
- The install from kickstart will continue until it reaches the point in which a password for the system is need. Once you enter the root password for the system, the install will continue to formatting the hard-drive(s). Then the installation of packages will start briefly, execute any post scripts and finally reboot.
- Now that the system has an initial installation with a core set of packages, we can now continue to registering the system to RHN or Satellite. Since each of the cluster nodes are remote, tunnelling X over ssh provides a way to work efficiently with the systems. Since you may be using ssh, it will be helpful to set-up a password-less login between the systems. This can be done with ssh-keygen and ssh-copy-id.
- The command `rhn_register` will start the RHN registration wizard. You will have the option to register your system to RHN or an available [RHN Satellite](#). You will be asked for an ID and a password for registration, and then you will need to complete the remainder of the steps to get your system registered. Here are a few screen shots of the registration process.

```
[root@db1 ~]# rhn_register
```

This step could also be automated by using the command `rhnreg_ks`. One option available with 'rhnreg\_ks' is to supply an activation key. The activation key needs to be set-up in advance on RHN, and this will register a system with the correct base channel, child channels, install additional packages, configurations files deployed and finally assign it to the correct system group. Here is an example of registration with an activation key.

```
[root@db1 ~]# rhnreg_ks --activationkey=youractivationkey
```



- Once each of the systems are registered, the following channels will need to be made available to the cluster nodes for updates and also additional packages.

```
rhel-x86_64-server-5  
rhel-x86_64-server-cluster-5  
rhel-x86_64-server-cluster-storage-5  
rhn-tools-rhel-x86_64-server-5
```

- Each system can be updated to the latest minor release of Red Hat Enterprise Linux with the following command. The systems should then be rebooted to be running on the latest kernel.

```
/usr/bin/yum upgrade -y; /usr/bin/reboot
```

- Now that each of the systems are running the latest release of the Red Hat distribution, we can move on to installing both the additional cluster software and the Caché database.



## 6.2 Software Installation - Cluster Software

In this section we will focus on installing the cluster software. There are a handful of packages that need to be installed. You can use the yum client tool to install the entire cluster and cluster-storage groups (/usr/bin/yum groupinstall Clustering 'Cluster Storage') or install the specific packages. Here is a list of the packages to be installed along with the command. The following packages should also install additional dependencies.

- **rgmanager**
- **cman**
- **ricci**
- **luci**

The luci package installs the cluster web interface server (known as Conga) that is used to build and manage the cluster. This package only needs to be installed on one system. The cluster web interface server can be run on one of the nodes within the cluster or a node outside of the cluster. The recommended approach is to run the cluster web interface server on a system separate from the running cluster.

```
[root@db1 ~]# /usr/bin/yum install cman ricci rmanager luci -y
Loaded plugins: rhnplugin, security
Setting up Install Process
No package rmanager available.
Resolving Dependencies
--> Running transaction check
--> Package cman.x86_64 0:2.0.115-85.el5_7.1 set to be updated
--> Processing Dependency: python-suds for package: cman
--> Processing Dependency: python-pycurl for package: cman
--> Processing Dependency: expect for package: cman
--> Processing Dependency: perl(XML::LibXML) for package: cman
--> Processing Dependency: perl(Net::Telnet) for package: cman
--> Processing Dependency: openais for package: cman
--> Processing Dependency: libcpkg.so.2(OPENAIS_CPG_1.0)(64bit) for package: cman
--> Processing Dependency: libSaCkpt.so.2(OPENAIS_CKPT_B.01.01)(64bit) for
package: cman
--> Processing Dependency: libcpkg.so.2()(64bit) for package: cman
--> Processing Dependency: libSaCkpt.so.2()(64bit) for package: cman
--> Package luci.x86_64 0:0.12.2-32.el5 set to be updated
--> Processing Dependency: python-imaging for package: luci
--> Package ricci.x86_64 0:0.12.2-32.el5 set to be updated
--> Processing Dependency: modcluster >= 0.12.0 for package: ricci
--> Processing Dependency: oddjob for package: ricci
--> Running transaction check
--> Package modcluster.x86_64 0:0.12.1-2.el5 set to be updated
--> Package oddjob.x86_64 0:0.27-11.el5 set to be updated
--> Processing Dependency: oddjob-libs = 0.27-11.el5 for package: oddjob
--> Processing Dependency: liboddjob.so.0()(64bit) for package: oddjob
--> Package openais.x86_64 0:0.80.6-30.el5_7.1 set to be updated
--> Package perl-Net-Telnet.noarch 0:3.03-5 set to be updated
--> Package perl-XML-LibXML.x86_64 0:1.58-6 set to be updated
--> Processing Dependency: perl-XML-SAX for package: perl-XML-LibXML
--> Processing Dependency: perl-XML-Namespacesupport for package: perl-XML-
LibXML
```



```

--> Processing Dependency: perl-XML-LibXML-Common for package: perl-XML-LibXML
--> Processing Dependency: perl(XML::SAX::Exception) for package: perl-XML-LibXML
--> Processing Dependency: perl(XML::SAX::DocumentLocator) for package: perl-XML-LibXML
--> Processing Dependency: perl(XML::SAX::Base) for package: perl-XML-LibXML
--> Processing Dependency: perl(XML::NamespaceSupport) for package: perl-XML-LibXML
--> Processing Dependency: perl(XML::LibXML::Common) for package: perl-XML-LibXML
---> Package pexpect.noarch 0:2.3-3.el5 set to be updated
---> Package python-imaging.x86_64 0:1.1.5-7.el5 set to be updated
--> Processing Dependency: tkinter for package: python-imaging
---> Package python-pycurl.x86_64 0:7.15.5.1-8.el5 set to be updated
---> Package python-suds.noarch 0:0.4.1-2.el5 set to be updated
--> Running transaction check
---> Package oddjob-libs.x86_64 0:0.27-11.el5 set to be updated
---> Package perl-XML-LibXML-Common.x86_64 0:0.13-8.2.2 set to be updated
---> Package perl-XML-NamespaceSupport.noarch 0:1.09-1.2.1 set to be updated
---> Package perl-XML-SAX.noarch 0:0.14-8 set to be updated
---> Package tkinter.x86_64 0:2.4.3-44.el5 set to be updated
--> Processing Dependency: libTix8.4.so()(64bit) for package: tkinter
--> Running transaction check
---> Package tix.x86_64 1:8.4.0-11.fc6 set to be updated
--> Finished Dependency Resolution

```

#### Dependencies Resolved

Package	Arch	Version	Repository	Size
<b>Installing:</b>				
cman	x86_64	2.0.115-85.el5_7.1	rhel-x86_64-server-5	762 k
luci	x86_64	0.12.2-32.el5	rhel-x86_64-server-cluster-5	26 M
ricci	x86_64	0.12.2-32.el5	rhel-x86_64-server-cluster-5	1.2 M
<b>Installing for dependencies:</b>				
modcluster	x86_64	0.12.1-2.el5	rhel-x86_64-server-cluster-5	333 k
oddjob	x86_64	0.27-11.el5	rhel-x86_64-server-5	61 k
oddjob-libs	x86_64	0.27-11.el5	rhel-x86_64-server-5	44 k
openais	x86_64	0.80.6-30.el5_7.1	rhel-x86_64-server-5	402 k
perl-Net-Telnet	noarch	3.03-5	rhel-x86_64-server-5	56 k
perl-XML-LibXML	x86_64	1.58-6	rhel-x86_64-server-5	229 k
perl-XML-LibXML-Common	x86_64	0.13-8.2.2	rhel-x86_64-server-5	16 k
perl-XML-NamespaceSupport	noarch	1.09-1.2.1	rhel-x86_64-server-5	15 k
perl-XML-SAX	noarch	0.14-8	rhel-x86_64-server-5	77 k
pexpect	noarch	2.3-3.el5	rhel-x86_64-server-5	214 k
python-imaging	x86_64	1.1.5-7.el5	rhel-x86_64-server-5	406 k
python-pycurl	x86_64	7.15.5.1-8.el5	rhel-x86_64-server-5	73 k
python-suds	noarch	0.4.1-2.el5	rhel-x86_64-server-5	251 k
tix	x86_64	1:8.4.0-11.fc6	rhel-x86_64-server-5	333 k
tkinter	x86_64	2.4.3-44.el5	rhel-x86_64-server-5	280 k

#### Transaction Summary

```

=====
Install      18 Package(s)
Upgrade      0 Package(s)

```



Total download size: 30 M

Downloading Packages:

(1/18): perl-XML-NamespaceSupport-1.09-1.2.1.noarch.rpm	15 kB	00:00
(2/18): perl-XML-LibXML-Common-0.13-8.2.2.x86_64.rpm	16 kB	00:00
(3/18): oddjob-libs-0.27-11.el5.x86_64.rpm	44 kB	00:00
(4/18): perl-Net-Telnet-3.03-5.noarch.rpm	56 kB	00:00
(5/18): oddjob-0.27-11.el5.x86_64.rpm	61 kB	00:00
(6/18): python-pycurl-7.15.5.1-8.el5.x86_64.rpm	73 kB	00:00
(7/18): perl-XML-SAX-0.14-8.noarch.rpm	77 kB	00:00
(8/18): pexpect-2.3-3.el5.noarch.rpm	214 kB	00:00
(9/18): perl-XML-LibXML-1.58-6.x86_64.rpm	229 kB	00:00
(10/18): python-suds-0.4.1-2.el5.noarch.rpm	251 kB	00:00
(11/18): tkinter-2.4.3-44.el5.x86_64.rpm	280 kB	00:00
(12/18): modcluster-0.12.1-2.el5.x86_64.rpm	333 kB	00:00
(13/18): tix-8.4.0-11.fc6.x86_64.rpm	333 kB	00:00
(14/18): openais-0.80.6-30.el5_7.1.x86_64.rpm	402 kB	00:00
(15/18): python-imaging-1.1.5-7.el5.x86_64.rpm	406 kB	00:00
(16/18): cman-2.0.115-85.el5_7.1.x86_64.rpm	762 kB	00:01
(17/18): ricci-0.12.2-32.el5.x86_64.rpm	1.2 MB	00:02
(18/18): luci-0.12.2-32.el5.x86_64.rpm	26 MB	00:25

-----  
Total 608 kB/s | 30 MB 00:51

Running rpm\_check\_debug

Running Transaction Test

Finished Transaction Test

Transaction Test Succeeded

Running Transaction

Installing	: perl-XML-LibXML-Common	1/18
Installing	: tix	2/18
Installing	: tkinter	3/18
Installing	: python-imaging	4/18
Installing	: openais	5/18
Installing	: python-pycurl	6/18
Installing	: perl-XML-NamespaceSupport	7/18
Installing	: perl-XML-SAX	8/18
Installing	: perl-XML-LibXML	9/18
Installing	: python-suds	10/18
Installing	: perl-Net-Telnet	11/18
Installing	: pexpect	12/18
Installing	: cman	13/18
Installing	: luci	14/18
Installing	: oddjob	15/18
Installing	: modcluster	16/18
Installing	: ricci	17/18
Installing	: oddjob-libs	18/18

Installed:

cman.x86_64 0:2.0.115-85.el5_7.1	luci.x86_64 0:0.12.2-32.el5
ricci.x86_64 0:0.12.2-32.el5	

Dependency Installed:

modcluster.x86_64 0:0.12.1-2.el5
oddjob.x86_64 0:0.27-11.el5
oddjob-libs.x86_64 0:0.27-11.el5
openais.x86_64 0:0.80.6-30.el5_7.1
perl-Net-Telnet.noarch 0:3.03-5



```
perl-XML-LibXML.x86_64 0:1.58-6
perl-XML-LibXML-Common.x86_64 0:0.13-8.2.2
perl-XML-Namespacesupport.noarch 0:1.09-1.2.1
perl-XML-SAX.noarch 0:0.14-8
pexpect.noarch 0:2.3-3.e15
python-imaging.x86_64 0:1.1.5-7.e15
python-pycurl.x86_64 0:7.15.5.1-8.e15
python-suds.noarch 0:0.4.1-2.e15
tix.x86_64 1:8.4.0-11.fc6
tkinter.x86_64 0:2.4.3-44.e15
```

Complete!

In addition to the main clustering packages listed above, there is an additional requirement for InterSystems Caché®'s only to be accessible by one node at a time. The cluster will be using HA-LVM to ensure that only one cluster node can access the shared volume. The second example provides the command and output for installing lvm2-cluster, which is needed for configuring the newest approach to HA-LVM.

- **lvm2-cluster**

The approach to configuring highly available lvm has been in use for many years, and recently a second option to providing HALVM is now provided. The original steps to configuring HA-LVM included a requirement for tagging lvm and creating a new initrd. The latest supported method no longer includes these requirements. Both the original and latest method are provided in this [knowledgebase document](#).

```
[root@db1 ~]# /usr/bin/yum install lvm2-cluster
Loaded plugins: rhnplugin, security
rhel-x86_64-server-cluster-storage-5 | 1.4 kB 00:00
rhel-x86_64-server-cluster-storage-5/primary | 26 kB 00:00
rhel-x86_64-server-cluster-storage-5 185/185
Setting up Install Process
Resolving Dependencies
--> Running transaction check
---> Package lvm2-cluster.x86_64 0:2.02.84-6.e15 set to be updated
--> Finished Dependency Resolution

Dependencies Resolved

=====
Package Arch Version Repository Size
=====
Installing:
lvm2-cluster x86_64 2.02.84-6.e15 rhel-x86_64-server-cluster-storage-5 284 k

Transaction Summary
=====
Install 1 Package(s)
Upgrade 0 Package(s)

Total download size: 284 k
Is this ok [y/N]: y
Downloading Packages:
lvm2-cluster-2.02.84-6.e15.x86_64.rpm | 284 kB 00:00
Running rpm_check_debug
```



```
Running Transaction Test
Finished Transaction Test
Transaction Test Succeeded
Running Transaction
  Installing      : lvm2-cluster                               1/1

Installed:
  lvm2-cluster.x86_64 0:2.02.84-6.el5

Complete!
```

## 6.3 Software Installation - Caché Database

Now that we have installed the operating system and updated it to the latest service pack level (minor release) and installed the necessary cluster software, the next step will be to install the Caché Database on to each system.

Caché can be installed by using the rpm approach, but Intersystems recommends for production deployments to use the 'cinstall' script. The script is included as a part of the normal download package, 2011.1.2.701.0su\_lnxrh5x64.tar.gz, and is available from Intersystems. Each system will need the package downloaded and uncompressed to a temporary directory for later configuration step.

```
[root@db1 ~]# mkdir /tmp/cache;cd /tmp/cache

[root@db1 cache]# tar xzvf 2011.1.2.701.0su_lnxrh5x64.tar.gz

[root@db1 cache]# ls
2011.1.2.701.0su_lnxrh5x64.tar.gz  copyright.pdf  docs          LICENSE      tools
cinstall                          cplatname     kitlist       NOTICE
cinstall_client                   dist           lgpl.txt      package
```

Additionally, here is the rpm method for installing Caché.

If you have set-up a custom repository using createrepo or your custom rpms are stored within a custom channel on a satellite, the first step will be to ensure that each of the systems can access the repository or channel that contains the package. Below is an example of a client repository configuration file that is located in '/etc/yum.repos.d/' and points to the hosted caché-server rpm that needs to be installed.

It is also good practice to sign the custom rpm that will be distributed. Listed below are a few quick steps to create a private and public pair key pair that can then be used to sign the rpm. This new public key will then need to be imported on to each system. After the rpm is signed and hosted centrally, then '/usr/bin/yum install caché-server -y' can be used to install the signed package.

```
## rpmuser created a private/public
[rpmuser@db1 ~]$ /usr/bin/gpg --gen-key

## rpmuser references available keys
[rpmuser@db1 ~]$ /usr/bin/gpg --list-keys

## rpmuser prepares an ~/.rpmmacros configuration file
[rpmuser@db1 ~]$ cat .rpmmacros
%_signature      gpg
```



```
%_gpg_name      RPM User

## rpmuser signs the cache-server package
[rpmuser@db1 ~]$ rpm --addsign cache-server-2011.1.0.532.0su-1.rh5.x86_64.rpm
Enter pass phrase:
Pass phrase is good.
cache-server-2011.1.0.532.0su-1.rh5.x86_64.rpm:

## rpmuser can resign the package
[rpmuser@db1 ~]$ rpm --resign cache-server-2011.1.0.532.0su-1.rh5.x86_64.rpm
Enter pass phrase:
Pass phrase is good.
cache-server-2011.1.0.532.0su-1.rh5.x86_64.rpm:
```

The example provides the commands for pushing an rpm to an already created satellite channel. There is also a listing of commands and configs that are needed to create a repo on the web server and the additional yum client configuration file.

```
## The rpmuser pushing packages to a managed channel on satellite
[rpmuser@db1 ~]$ /usr/bin/rhnpush -c InterSystems Caché@-5-x86_64 -u
channeladmin ~/cache-server-2011.1.0.532.0su-1.rh5.x86_64.rpm

## Root creating a repo in the directory that contains the target hosted rpms.
[root@lab ~]# /usr/bin/createrepo --checksum sha
/var/www/html/pub/distro/vendorApps

## The configuration repo file residing on each of the cluster nodes
[root@db1 ~]# cat /etc/yum.repos.d/vendorApps.repo
[vendorApps]
name=vendorApps
baseurl=http://lab.privnet.redhat.com/pub/distro/vendorApps
enabled=1
gpgcheck=1
```

This example is using yum to install the Caché server rpm. If you are centrally managing the Caché server rpm, then you can install the Caché server package with yum as well. Lastly, the installation of both the Red Hat software and vendor applications can also be deployed during the initial kickstart of the system.

```
##Installing the rpm package from a configured remote repository.
[root@db1 ~]# /usr/bin/yum install caché-server -y

##Installing the rpm package from a local source.
[root@db1 ~]# /usr/bin/yum localinstall ~/caché-server-2011.1.0.532.0su-
1.rh5.x86_64.rpm -y
```

## 6.4 Software Installation - Summary

At this point each server should have the correct software installed and updated to the latest minor release of Red Hat Enterprise Linux 5. Our focus in the next chapter will turn to configuring the systems, so each is ready to be clustered. This will include such items as configuring bonding, multipath, storage, other cluster resources and the cluster configuration itself.



# 7 Software Configuration

In this chapter the focus will be configuring each of the systems for redundancy and together as a working high availability cluster. This will include configuring multiple network bonds to isolate cluster and client traffic to its own network, the storage network, multipath, quorum disk, the cluster, and the HA-LVM clustered volume. The chapter will conclude with walking through the set-up of the InterSystems Caché database as a clustered service.

## 7.1 Software Configuration - Network Bonding

As an additional way to increase the tolerance of hardware failures Red Hat Enterprise Linux has the ability to bond network cards together in order to weather hardware failures. The following are the steps to configure network bonding for two networks (cluster network and client network).

The first step will be to modify the `modprobe.conf` file to include an alias between the network bonds and bonding modules. Additional options should also be included such as the bonding mode (mode=1 is for active-backup). Additional information about the bonding module can be reviewed with `'modinfo -p bonding'`. The following output should provide guidance in setting up your `modprobe` file. After modifying the `modprobe.conf` file, then the bonding module needs to be added with `modprobe` to the running kernel.

```
### The additions to modprobe.conf
[root@db1 ~]# cat /etc/modprobe.conf |grep -i bond
alias bond0 bonding
options bond0 mode=0 miimon=100 use_carrier=0
alias bond1 bonding
options bond1 mode=1 miimon=100 use_carrier=0

### Insert the bonding module to the running kernel.
[root@db1 ~]# /sbin/modprobe bonding

### Check that the bonding module is now in place and ready to be used.
[root@db1 ~]# lsmod |grep bond
bonding                142537  0
```

If you have not already done so, each systems' hosts file should be updated as a precaution to DNS failing. The following is the output from one systems hosts file.

```
[root@db1 ~]# cat /etc/hosts
# Do not remove the following line, or various programs
# that require network functionality will fail.
127.0.0.1        localhost.localdomain localhost
#:::1          localhost6.localdomain6 localhost6
172.31.224.254  lab.privnet.redhat.com lab
172.31.224.61   db1.privnet.redhat.com db1
172.31.224.62   db2.privnet.redhat.com db2
192.168.0.61    priv1.clunet.redhat.com  priv1
192.168.0.62    priv2.clunet.redhat.com  priv2
172.16.0.61     stor1.clunet.redhat.com  stor1
172.16.0.62     stor2.clunet.redhat.com  stor2
```



Next will be to create or modify the bonding and network configuration files in the network-scripts directory. The bonds will be considered master and the appropriate network interface cards (ifcfg-eth) will become the slaves. The output of the creation and modification of the files follow and include output for both the client and private (cluster) networks.

```
### db1's output of the created or modified network-script files.
[root@db1 network-scripts]# more /etc/sysconfig/network-scripts/ifcfg-*
::::::::::::
/etc/sysconfig/network-scripts/ifcfg-bond0
::::::::::::
### Client Network Bond
DEVICE=bond0
IPADDR=172.31.224.61
NETMASK=255.255.255.0
NETWORK=172.31.224.0
ONBOOT=yes
BOOTPROTO=static
USERCTL=no

::::::::::::
/etc/sysconfig/network-scripts/ifcfg-bond1
::::::::::::
### Private Cluster Network Bond
DEVICE=bond1
IPADDR=192.168.0.61
NETMASK=255.255.255.0
NETWORK=192.168.100.0
ONBOOT=yes
BOOTPROTO=static
USERCTL=no

::::::::::::
/etc/sysconfig/network-scripts/ifcfg-eth0
::::::::::::
# Virtio Network Device
DEVICE=eth0
USERCTL=no
BOOTPROTO=none
HWADDR=52:54:00:DB:BF:22
ONBOOT=yes
MASTER=bond0
SLAVE=yes

::::::::::::
/etc/sysconfig/network-scripts/ifcfg-eth1
::::::::::::
# Virtio Network Device
DEVICE=eth1
HWADDR=52:54:00:be:1e:94
ONBOOT=yes
USERCTL=no
MASTER=bond0
SLAVE=yes
BOOTPROTO=none

::::::::::::
/etc/sysconfig/network-scripts/ifcfg-eth2
```



```
.....  
# Virtio Network Device  
DEVICE=eth2  
HWADDR=52:54:00:AA:D1:A9  
ONBOOT=yes  
USERCTL=no  
MASTER=bond1  
SLAVE=yes  
BOOTPROTO=none  
  
.....  
/etc/sysconfig/network-scripts/ifcfg-eth3  
.....  
# Virtio Network Device  
DEVICE=eth3  
HWADDR=52:54:00:55:db:f1  
ONBOOT=yes  
MASTER=bond1  
SLAVE=yes  
BOOTPROTO=none  
USERCTL=no
```

```
[root@db1 ~]# ifconfig |grep -i -A 2 bond  
bond0      Link encap:Ethernet  HWaddr 52:54:00:DB:BF:22  
            inet addr:172.31.224.61  Bcast:172.31.224.255  Mask:255.255.255.0  
            inet6 addr: fe80::5054:ff:fedb:bf22/64 Scope:Link  
--  
bond1      Link encap:Ethernet  HWaddr 52:54:00:AA:D1:A9  
            inet addr:192.168.0.61  Bcast:192.168.0.255  Mask:255.255.255.0  
            inet6 addr: fe80::5054:ff:feaa:d1a9/64 Scope:Link
```

### ### db2's output of the created or modified network-script files.

```
/etc/sysconfig/network-scripts/ifcfg-bond0  
.....  
DEVICE=bond0  
IPADDR=172.31.224.62  
NETMASK=255.255.255.0  
NETWORK=172.31.224.0  
ONBOOT=yes  
BOOTPROTO=static  
USERCTL=no  
  
.....  
/etc/sysconfig/network-scripts/ifcfg-bond1  
.....  
DEVICE=bond1  
IPADDR=192.168.0.62  
NETMASK=255.255.255.0  
NETWORK=192.168.0.0  
ONBOOT=yes  
BOOTPROTO=static  
USERCTL=no  
  
.....  
/etc/sysconfig/network-scripts/ifcfg-eth0
```



```
.....  
# Virtio Network Device  
DEVICE=eth0  
USERCTL=no  
BOOTPROTO=none  
HWADDR=52:54:00:CB:FA:92  
ONBOOT=yes  
MASTER=bond0  
SLAVE=yes  
  
.....  
/etc/sysconfig/network-scripts/ifcfg-eth1  
.....  
# Virtio Network Device  
DEVICE=eth1  
USERCTL=no  
ONBOOT=yes  
MASTER=bond0  
SLAVE=yes  
BOOTPROTO=none  
HWADDR=52:54:00:3C:84:EF  
  
.....  
/etc/sysconfig/network-scripts/ifcfg-eth2  
.....  
# Virtio Network Device  
DEVICE=eth2  
USERCTL=no  
HWADDR=52:54:00:46:59:47  
ONBOOT=yes  
MASTER=bond1  
SLAVE=yes  
BOOTPROTO=none  
  
.....  
/etc/sysconfig/network-scripts/ifcfg-eth3  
.....  
# Virtio Network Device  
DEVICE=eth3  
USERCTL=no  
MASTER=bond1  
ONBOOT=yes  
SLAVE=yes  
BOOTPROTO=none  
HWADDR=52:54:00:0C:D2:01
```

**### The output of ifconfig once the network bonds are operational.**

```
[root@db2 ~]# ifconfig |grep -i -A 2 bond  
bond0      Link encap:Ethernet  HWaddr 52:54:00:CB:FA:92  
           inet addr:172.31.224.62  Bcast:172.31.224.255  Mask:255.255.255.0  
           inet6 addr: fe80::5054:ff:feeb:fa92/64 Scope:Link  
           --  
bond1      Link encap:Ethernet  HWaddr 52:54:00:46:59:47  
           inet addr:192.168.0.62  Bcast:192.168.0.255  Mask:255.255.255.0  
           inet6 addr: fe80::5054:ff:fe46:5947/64 Scope:Link
```



## 7.2 Software Configuration - The Storage Network

While network bonds provide a level of protection against network card failures for our client and cluster networks, we will configure a third set of networks specifically for connecting to an iscsi-target over tcp. In the examples below, we will work through using iscsi client tools to configure the storage network. If you are using fibre storage instead of iscsi storage, you will probably want to skip this section and move to configuring multipath.

The first step will be to configure our network cards that will be used for the storage network. In the case of the example below, it is the fifth (eth4) and sixth (eth5) interface card in each system. Each is configured with a static network of 172.16.0.0/24. For additional redundancy each connection could also be configured to travel over a different switch or different network.

```
## The output of system db1 4th and 5th network cards.
```

```
.....  
/etc/sysconfig/network-scripts/ifcfg-eth4  
.....  
# Virtio Network Device  
DEVICE=eth4  
BOOTPROTO=static  
ONBOOT=yes  
HWADDR=52:54:00:20:c9:af  
USERCTL=no  
IPADDR=172.16.0.51  
NETMASK=255.255.255.0  
.....  
/etc/sysconfig/network-scripts/ifcfg-eth5  
.....  
# Virtio Network Device  
DEVICE=eth5  
BOOTPROTO=static  
ONBOOT=yes  
HWADDR=52:54:00:7e:2a:91  
USERCTL=no  
IPADDR=172.16.0.61  
NETMASK=255.255.255.0
```

```
## The output of system db2 4th and 5th network cards.
```

```
.....  
/etc/sysconfig/network-scripts/ifcfg-eth4  
.....  
# Virtio Network Device  
DEVICE=eth4  
BOOTPROTO=static  
ONBOOT=yes  
HWADDR=52:54:00:3e:d3:28  
USERCTL=no  
#MASTER=bond2  
#SLAVE=yes  
IPADDR=172.16.0.52  
NETMASK=255.255.255.0  
.....  
/etc/sysconfig/network-scripts/ifcfg-eth5  
.....  
# Virtio Network Device
```



```
DEVICE=eth5
BOOTPROTO=static
ONBOOT=yes
HWADDR=52:54:00:2e:e2:fc
USERCTL=no
#MASTER=bond2
#SLAVE=yes
IPADDR=172.16.0.62
NETMASK=255.255.255.0
```

Now that each system's network cards are configured and can ping the storage, the next step will be to configure each host as an iscsi initiator (initiates connections to the iscsi storage target). Once the systems are configured, the systems will automatically connect to the target on boot and see the storage. The iscsi target in this case is a Red Hat Enterprise Linux 6 server serving out luns using the included iscsi-target software. While the target is outside the scope of this document, the configuration and steps are included below for reference.

```
## Installing the scsi-target package
[root@lab ~]# /usr/bin/yum install scsi-target-utils -y

## Configuring the Firewall to allow connections from the initiators
[root@lab ~]# iptables -I INPUT 3 -s 172.16.0.0/24 -p tcp --dport 3260 -j ACCEPT
[root@lab ~]# /sbin/service iptables save;/sbin/service iptables restart

## The target configuration shares out a single lun which is backed by a logical volume.
[root@lab ~]# cat /etc/tgt/targets.conf |grep -i stornet -A8
<target iqn.2011-09.com.redhat.stornet:server.target1>
    direct-store /dev/libvirtpool/lun1
</target>

##db1 eth[45]
initiator-address 172.16.0.51
initiator-address 172.16.0.61
##db2 eth[45]
initiator-address 172.16.0.52
initiator-address 172.16.0.62

## Starting and persisting the target service to be enabled on boot.
[root@lab ~]# /sbin/service tgtd start;/sbin/chkconfig tgtd on;
```

Now that the target is configured we can modify each system's iscsi iface device files and to initiate a persistent connection to the storage over both of its network devices. The iscsiadm command line tool will do the heavy lifting for this task. The examples that follow provide the commands and output of the iscsi files that were used to configure each of the four connections.

The first step is to configure iscsi iface device files by creating and updating a file for each network that will connect to the storage. The two commands that need to be run for each connection are the following. You can also edit the files by hand as needed.

```
## Example commands that were run on db1
[root@db2 ~]# iscsiadm -m iface -I iface0 --op=new
[root@db2 ~]# iscsiadm -m iface -I iface1 --op=new
```



```
[root@db2 ~]# iscsiadm -m iface -I iface0 --op=update -n iface.hwaddress 52:54:00:20:C9:AF -n iface.ipaddress 172.16.0.51
[root@db2 ~]# iscsiadm -m iface -I iface1 --op=update -n iface.hwaddress 52:54:00:7E:2A:91 -n iface.ipaddress 172.16.0.61
```

#### ## Example commands that were run on db2

```
[root@db2 ~]# iscsiadm -m iface -I iface0 --op=new
[root@db2 ~]# iscsiadm -m iface -I iface1 --op=new
[root@db2 ~]# iscsiadm -m iface -I iface0 --op=update -n iface.hwaddress 52:54:00:3E:D3:28 -n iface.ipaddress 172.16.0.52
[root@db2 ~]# iscsiadm -m iface -I iface1 --op=update -n iface.hwaddress 52:54:00:2E:E2:FC -n iface.ipaddress 172.16.0.62
```

#### ## Here is the output of the iface configuration files

```
[root@db1 ~]# more /var/lib/iscsi/ifaces/iface*
```

```
.....:
/var/lib/iscsi/ifaces/iface0
.....:
iface.iscsi_ifacename = iface0
iface.hardware_address = 52:54:00:20:C9:AF
iface.ip.address = 172.16.0.51
iface.transport_name = tcp
```

```
.....:
/var/lib/iscsi/ifaces/iface1
.....:
iface.iscsi_ifacename = iface1
iface.hardware_address = 52:54:00:7E:2A:91
iface.ip.address = 172.16.0.61
iface.transport_name = tcp
```

```
[root@db2 ~]# more /var/lib/iscsi/ifaces/iface*
```

```
.....:
/var/lib/iscsi/ifaces/iface0
.....:
# BEGIN RECORD 2.0-872
iface.iscsi_ifacename = iface0
iface.hardware_address = 52:54:00:3E:D3:28
iface.ip.address = 172.16.0.52
iface.transport_name = tcp
# END RECORD
```

```
.....:
/var/lib/iscsi/ifaces/iface1
.....:
# BEGIN RECORD 2.0-872
iface.iscsi_ifacename = iface1
iface.hardware_address = 52:54:00:2E:E2:FC
iface.ip_address = 172.16.0.62
iface.transport_name = tcp
# END RECORD
```

Now that the iscsi interfaces are configured the next step will be to start and persist the iscsi daemon. Once the daemon is running the last steps will be to discover and store your connection information to the new storage.

```
## Persisting and starting the iscsi service
```



```
[root@db2 ~]# /sbin/service iscsi start;/sbin/chkconfig iscsi on

## Discovering the target storage
[root@db2 ~]# iscsiadm -m discoverydb -t st -p 172.16.0.254 -I iface0 -I
iface1 --discover

## Storing and logging in to the target storage
[root@db2 ~]# iscsiadm -m node --targetname iqn.2011-
09.com.redhat.stornet:server.target1 -p 172.16.0.254:3260 --login
```

Each server should now have access to the network based storage and most likely seeing new available disks. Now that we have multiple connections and multiple disks showing up, we should now move on to configuring our multipath software.

## 7.3 Software Configuration - Red Hat Enterprise Linux Multipath

When you have redundant fibre or iscsi connections configured to your system, your scsi disk information may multiply. The multipath software included in Red Hat Enterprise Linux provides management of these scsi devices to enable a single multipath (mpath) device. The multipath software additionally manages traffic over redundant paths to ensure consistent access to storage. In order to use multipath a few steps will need to be followed for each system.

First we will need to check that the device-mapper-multipath package is installed. The yum command can help use with this task.

```
[root@db1 network-scripts]# yum list installed |grep multipath
device-mapper-multipath.x86_64                0.4.7-46.el5_7.1                installed
```

Now that we know multipath is installed, we will need to configure the multipath configuration file on each system. At the very least you should comment out the blacklist section as shown below, and then make additional modifications to meet your requirements. You will want to replicate the changes to the other systems in the cluster. Here is the output of the configuration from db1.

```
#blacklist {
#     devnode "*"
#}

[root@db1 ~]# cat /etc/multipath.conf
defaults {
udev_dir                /dev
polling_interval        10
selector                "round-robin 0"
path_grouping_policy    multibus
getuid_callout          "/sbin/scsi_id -g -u -s /block/%n"
prio_callout            none
path_checker            readsector0
rr_min_io               100
max_fds                 8192
rr_weight               priorities
failback                immediate
```



```
no_path_retry          fail
user_friendly_names   yes
}
```

Once the configuration file is modified, then the multipath service will need to be started and set to be persistent across reboots. Here is the command to start and persist the service. Once the service is up and running, you should test that everything is configured as expected. The multipath command line interface will be helpful for testing and reviewing your multipath set-up.

```
##Persisting the multipath service
[root@db1 network-scripts]# /sbin/service multipathd start;/sbin/chkconfig
multipathd on

[root@db1 ~]# multipath -ll
mpath0 (SATA_WDC_WD3200BEKT-_WD-WX31A70H3425) dm-1 ATA,WDC WD3200BEKT-0
[size=10G][features=0][hwhandler=0][rw]
\_ round-robin 0 [prio=2][active]
  \_ 3:0:0:1 sdb 8:16 [active][ready]
  \_ 2:0:0:1 sda 8:0 [active][ready]
```

At this point in the configuration our network and storage connections should be fully redundant and functioning properly. We should also be able to access our shared storage.

## 7.4 Software Configuration - Prepare Cluster Environment

At this point the required cluster software should be installed, networking and storage configured for redundancy. The next step in the process will be to configure the firewall and review active services.

- The cluster will communicate over a number of network ports, and it is important that the cluster communications are not blocked. See the [Red Hat Cluster Administration Guide](#) for configuring the firewall to allow relevant cluster and migration traffic between hosts.
- It is also good practice to disable any non-essential services for security reasons but also to reduce start-up times of cluster nodes.

## 7.5 Software Configuration - The Cluster

Now that the hardware and software requirements have been met, software installed and each of the systems configured and readied as cluster nodes, the work of building a cluster can begin.

### RICCI Service

- First, the ricci service used for authentication within the cluster will need to be started and persisted across reboots for each cluster node.

```
[root@db1 network-scripts]# /sbin/service ricci start;/sbin/chkconfig ricci
on
Starting oddjobd: [ OK ]
generating SSL certificates... done
```



Starting ricci: [ OK ]

## LUCI Service

- Next, a cluster web service called luci will be used for the initial configuration of the cluster. This service can be installed on one of the local nodes or optionally be run on a system outside of the cluster. The luci package will need to be installed as it contains the web-ui contents and scripts. For the reference build, we will be running the luci server on db1.
- The cluster manager instance can be created by running the luci\_admin command as root. It will prompt for a password, and then provide a web address of the manager. The web address for example will use secure socket layers and be bound to port 8084 such as 'https://db1.privnet.redhat.com:8084'.

```
[root@db1 network-scripts]# luci_admin init
Initializing the luci server

Creating the 'admin' user

Enter password:
Confirm password:

Please wait...
The admin password has been successfully set.
Generating SSL certificates...
The luci server has been successfully initialized

You must restart the luci server for changes to take effect.

Run "service luci restart" to do so

[root@db1 network-scripts]# /sbin/service luci restart;/sbin/chkconfig luci
on
Shutting down luci: [ OK ]
Starting luci: Generating https SSL certificates... done [ OK ]

Point your web browser to https://db1.privnet.redhat.com:8084 to access
luci
```

## Creating and Configuring the Cluster Nodes

Once the luci service is enabled and you have successfully logged in to the Conga (luci) Cluster Manager, there will be a few general steps to creating and configuring the hosts as a quorate cluster.

- The first step will be to create a new cluster. As shown in the figure, select the 'Create a New Cluster' button on the cluster tab. A cluster name will need to be provided. Then add the private network ip or private network hostname of each system that will perform as a cluster node. You will also need to include the root password of each



host.

The password will only be used for the initial adding a node and configuration. All authentication after cluster creation will occur through the ricci mechanism. For additional reading about cluster creation and the ricci protocol see the Red Hat Cluster Administration Guide.

- As long as the system is connected to Red Hat Network or Red Hat Satellite selecting '*Download packages*' will install any missing cluster rpms. You will want to deselect '*Enable Shared Storage Support*'. You will also want to select the '*Reboot nodes before joining cluster*' option. Once the form is filled out and submitted, the cluster manager will perform a number of steps to get the cluster nodes configured and active as a running quorate cluster.

Over the next few minutes you will notice the cluster downloading any required packages, creating a minimal cluster configuration (`/etc/cluster/cluster.conf`) for the cluster nodes and rebooting each of the systems.

Once each of the nodes has rebooted, a quorate cluster with two nodes on-line should be available. You can verify the cluster status either by viewing the status within Conga or by logging in to one of the systems and running the `clustat` command. If you run into problems during the creation of your cluster, you may want to check that the correct ports are open on each nodes firewall, there are no issues with selinux and the ricci daemon is enabled on each cluster node.

## Configuring Fence Devices

Now that the cluster is on-line and quorate, you will need to configure a few more items. First, in case there is an issue with one of the cluster nodes, an automated way to remove the cluster node from causing harm needs to be configured. This is normally done through a fencing device. There are a number of popular fence mechanisms such as integrated system fencing, shared power fencing, scsi-pr fencing and fabric fencing. In this example we are using a virtual fence device for our virtual cluster and scsi fence for our back-up fence device.

We will configure our fencing devices using the conga web interface, and the following are the steps to create and configure the mechanisms.

- First we will need to log-in to the web interface and click on the '*Cluster*' tab and click on the link representing the cluster. In the case of the reference build, we will click on the link '*cachedb1*'.
- Once located on the cluster configuration screen you will then click on the '*Shared Fence Devices*' link on the lower left hand corner of the screen.
- Then click on the '*Add a Fence Device*' button and choose the '*Virtual Machine Fencing*' option from the drop-down menu.
- You will then need to provide a '*Name*' for the Virtual Fence Device and click on the '*Add this shared fence device*' button to submit the form.

Since we will be using a back-up fence device, the next step in the process will be to create a shared fence device that each of the nodes can use for backup.



In this section we will be using scsi fencing, and there are a number of requirements to be met and can be referenced in the following document in section titled [SCSI Fencing Requirements and Limitations](#).

- In order to create the second device and in this case a 'SCSI Fencing' device, we first need to click on the 'Add a Fence Device' button and choose the 'SCSI Fencing' option from the drop-down menu.
- Then on the 'Add a Sharable Fence Device' screen 'Fence Type' should be 'SCSI Fencing', and you will need to provide a 'Name' for the scsi fence mechanism. In the reference build it is called 'scsifence'.
- Now that the form for the shared scsi fence device is filled in completely, submit the form by clicking the 'Add this shared fence device' button.
- Now under the 'Shared Fence Devices' you should have at a minimum two devices, 'vfence' and 'scsifence'.
- Now these devices will need to be associated to each of the nodes in the cluster by clicking the nodes link, and then choosing 'Manage Fencing for this Node'.
- Next you will need to click 'Add a fence device to this level' and select the existing shared fencing device from the drop-down. In the case of the reference build the primary fencing will be virtual fencing previously configured, and we will select 'vfence' from the list of existing fence devices. Then we will provide a domain that should be assigned to 'vfence' which will be *db1* for node *priv1.clunet.redhat.com*.
- The final step to configuring the primary fence device will be to click the 'Update main fence properties'.

After the primary fencing is configured our attention will turn to configuring of the back-up fence device.

- This can be done by clicking on the 'Add a fence device to this level' under 'Backup Fencing Method' and select the existing shared fencing device from the drop-down.
- In the case of the reference build the backup fencing will be scsi fencing, and we will select 'scsifence' from the list of existing fence devices. Then we will provide a domain that should be assigned to the 'vfence' device, which will be *db1* for node *priv1.clunet.redhat.com* and finally click on the 'update backup fence properties' button for the node to complete the process.

## Configuring a Fail-Over Domain

Creating a failover domain for the cluster will be the next component to configure. The failover domain will be the logical group of systems that will be enabled to host the active highly available service. The steps to create a failover domain are listed below.

- The first step in the process to creating this domain will be to click on the 'Cluster' tab and click on the 'Failover Domains' link in the lower left hand corner of the screen.
- Next you will choose 'Add a Failover Domain' and provide a name for the new domain.
- The domain for the InterSystems Caché® database service will be restricted which can



be set by choosing the *'Restrict failover to this domain's members'* box.

- The InterSystems Caché® domain will be prioritized with services starting and running on cluster members in a pre-determined ordered manner. This can be set by choosing the *'Prioritized'* box.
- It is also recommended for the InterSystems Caché® domain to have services set to automatically not fail back. This recommendation can be set by choosing the *'Do not fail back services in this domain'* box.
- The last step in the domain configuration will be to select the cluster node members available in the domain by clicking the box next to each node. Each node will need to be weighted to represent the priority of the start order. A node with a priority of one will be used before a node with a priority of two.
- Now that the cluster domain options are selected the final step will be to click the *'Submit'* button. This last step will make the domain available in the cluster menu, so it can be used during the creation of services at a later time.

At this point, an operational cluster should be on-line. The status of the cluster can be viewed from within the web interface or by running `clustat` at the command line from one of the nodes.

## 7.6 Software Configuration - Quorum Disk

In this section we will be discussing and configuring the optional quorum disk. While the quorum disk is considered an optional add-on, it is typically recommended and considered a best practice to use the quorum disk in two node cluster configurations. The quorum disk will ward off any possible race conditions that may appear when a failure occurs. Please see the following [Red Hat reference guide](#) to learn more information about the quorum disk.

- The first step in the quorum process will be to allocate at least a 10MB shared lun to both of the nodes. The quorum disk could also be placed on almost any other shared storage device as long as each of the nodes in the cluster have access to it, and it can be mounted with `o_direct`.
- The next step will be to initialize the shared device as a quorum disk with `mkqdisk` as noted below. The `-c` creates the device and `-l` creates a label for the device. Now that the quorum disk is initialized and labelled, a few select options will need to be chosen.
- Quorum Disk can be configured by selecting the *'Cluster'* tab and then choosing the cluster name called *'cachedb1'* and click the *'Quorum Partition'* tab.
- Next select the radio button *'Use a Quorum Partition'*, and then provide the information for the quorum partition. In the case of the quorum partition an interval, votes, `tko` and minimum score are needed. The interval will determine how often to test. The number of votes will be the number represented to the rest of the cluster when quorum is being determined (Note. A cluster is considered quorate with 51% of the cluster nodes as represented by votes are on-line). For this solution each cluster node will get a single vote with the quorum partition also getting a single vote.
- In this reference, we will not be using a Heuristic and can be left empty. We will



however be using the *'master\_wins=1'* option and can be referenced below. This option wards off the possibility of a fence race by allowing the qdisk master to win. This option will be added to the cluster configuration in a later step using your favorite editor.

- Our next step will be to provide our quorum disk label as created in the previous step.

```
## Creating the quorum disk with a label called qdisk.
[root@db1 ~]# mkqdisk -c /dev/vdb -l qdisk
mkqdisk v0.6.0
Writing new quorum disk label 'qdisk' to /dev/vdb.
WARNING: About to destroy all data on /dev/vdb; proceed [N/y] ?

## A listing of the available quorum disks on db1.
[root@db1 ~]# mkqdisk -L
mkqdisk v0.6.0
/dev/vdb1:
  Magic:                eb7a62c2
  Label:                qdisk
  Created:              Mon Sep 12 18:51:31 2011
  Host:                 db1.privnet.redhat.com
  Kernel Sector Size:  512
  Recorded Sector Size: 512
```

The remaining steps to configuring the quorum partition as a part of this solution will be to modify the cluster configuration file and propagate the changes throughout the cluster. The changes to the `/etc/cluster/cluster.conf` file are as follows.

- Log-in to any cluster node as root and open the `/etc/cluster/cluster.conf` file with your favorite editor.
- At the top of the file there is a `config_version` for the cluster file. Increment it by one.
- Within the cluster file, there should be place that includes `total_votes`. This should be modified to say *'total\_votes=3'*. Additionally, a change will be made to *'two\_node=1'* to *'two\_node=0'*.
- Because we are using the `master_wins` option, we will make modifications to the `cman`, `totem` and `quorumd` lines as referenced below. Some of the changes are specific to timing during failure events. The changes to the `cman` and `totem` lines will include `totem_timeout` and `dev_poll_options` and are set in milliseconds and should be equal. Also the *'master\_wins="1"'* needs to be added to the `quorumd` line. The reference below includes the changes mentioned.
- Now that the changes have been made, the cluster config incremented and saved, the entire cluster will need to be updated with the changes. This can be done with the simple command, *'ccs\_tool update /etc/cluster/cluster.conf'*.

```
## Make sure to increment the config_version prior to propagating with the
ccs_tool.
<cluster alias="cachedb1" config_version="78" name="cachedb1">

## The addition of the quorum_dev_poll option and checking that
expected_votes is set to 3.
<cman quorum_dev_poll="59000" expected_votes="3"/>
```



```
## The token time should be equal to the quorum_dev_poll time
<totem consensus="4800" join="60" token="59000"
token_retransmits_before_loss_const="20"/>

## The addition of the master_wins option is included on the following
line.
<quorumd interval="4" label="qdisk" master_wins="1" reboot="0" tko="8"
tko_up="2" votes="1" loglevel="7" log_facility="local4"/>
```

When modifying a cluster configuration file directly using a text editor instead of relying on the web interface, it is important to increment the configuration version of the cluster file and propagate the file to all nodes in a cluster. This is normally done with the `ccs_tool` command.

The normal operation of the `clustat` command while using heuristics without the `master_wins` option enabled will show each cluster node with a Quorum Disk as Online. When however the `master_wins` option is set to enabled with `master_wins="1"` without any heuristics listed, the `clustat` output will show only the current quorum disk master as online. The non-quorum disk master node will list its Quorum Disk as Offline. This is considered normal operation when using the `master_wins` option.

## 7.7 Software Configuration - HA LVM

Now that the cluster is quorate with a working quorum disk, the focus will shift to enabling the cluster environment to support a clustered database with a requirement of high available lvm (HA-LVM). HALVM provides a way for each node to access the shared volume but in a restricted manner where only one node can be active with a volume at a time.

One of the first steps that we will need to work through is creating a physical volume from the shared multipath device. Since multipath is configured along with friendly names (multipath configuration), you should have a device such as `mpath0`. We will first want to create a partition on the disk and tag it with lvm.

```
[root@db2 nodes]# fdisk /dev/mapper/mpath0
```

```
The number of cylinders for this disk is set to 1305.
There is nothing wrong with that, but this is larger than 1024,
and could in certain setups cause problems with:
1) software that runs at boot time (e.g., old versions of LILO)
2) booting and partitioning software from other OSs
   (e.g., DOS FDISK, OS/2 FDISK)
```

```
Command (m for help): p
```

```
Disk /dev/mapper/mpath0: 10.7 GB, 10737418240 bytes
255 heads, 63 sectors/track, 1305 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
```

Device	Boot	Start	End	Blocks	Id	System
/dev/mapper/mpath0p1		1	1305	10482381	8e	Linux LVM

Before we move on to working with the creation of the volume, we need to enable cluster locking on each node in the cluster. This can be done with the `lvmconf` command. Additionally, you will want to start the ensure that the cluster logical volume daemon (`clvmd`) is running and



persisted again on each node. Also since we will be enabling cluster locking we will not need to worry about adding a `volume_list` tag for our cluster nodes in `lvm.conf` or creating new `initrd`. In the old method both of these steps were needed.

```
## Enable cluster locking
[root@db1 ~] /usr/sbin/lvmconf --enable-cluster

[root@db1 ~] /sbin/service clvmd start; /sbin/chkconfig clvmd on
```

Now that cluster locking is enabled, the next step in the process will be to create a physical volume from the partitioned `mpath` device. Once a physical volume is created we will continue with making the volume group, logical volume, formatting the new logical volume with the `ext3` filesystem and finally deactivation of the device.

The step that is worth a little more discussion will be the creation of the volume group. In the case below, we are creating the volume group with a cluster tag. While there are two approaches to HALVM, the preferred newer method is to use a clustered tag on the volume group. This will also leverage the additional cluster logical volume daemon, and this approach now adds additional protection against accidental administrative volume removal. The following steps were used to create the volumes, format and deactivate the device.

```
## Creation of the physical volume from the mpath device.
[root@db1 ~]# pvcreate /dev/mpath/mpath0p1

## Creation of a volume group with the clustered tag
[root@db2 nodes]# vgcreate -cy vgcache /dev/mpath/mpath0p1

## Creation of a logical volume
[root@db2 nodes]# lvcreate -l 100%VG -n lvcache vgcache

## Formatting the device with the ext3 filesystem
[root@db2 nodes]# mkfs.ext3 /dev/vgcache/lvcache

## Deactivating the logical volume
[root@db1 ~]# /usr/sbin/lvchange -an /dev/vgcache/lvcache
```

### The old HA-LVM steps for reference.

In the old method of HA-LVM, there were essentially four steps that needed to be taken on each system.

1. Create and format your device as described above, but do not tag the volume to be clustered (`vgcreate -cy`).
2. Add the volume and filesystem mount point to the cluster configuration such as `"lvm name="lvcache" vg_name="vgcache"`.
3. Then modify `/etc/lvm/lvm.conf` on each node (`@node_name` as found in `/etc/cluster/cluster.conf`).
4. The final step is to make an new `initrd`.



```
[root@db1 ~]# cman_tool status|egrep -i 'node name'
Node name: priv1.clunet.redhat.com

## edit the lvm.conf file to tag all local volumes and ha volume
as represented by @priv...

volume_list = [ "VolGroup00", "@priv1.clunet.redhat.com" ]

## A new initrd needs to be made so that the systems can see the
volumes
[root@db1 ~]# mkinitrd -f /boot/initrd$(uname -r).img $(uname -r)
```

The knowledgebase article titled [What is a Highly Available LVM \(HA-LVM\) configuration and how do I implement it?](#) is a good reference describing both the old and new approaches to HA-LVM

## 7.8 Software Configuration – InterSystems Caché as a Clustered Service

In this final section of creating a cluster service we will walk through each of the resources that the cluster will manage as a service. The resources that will make up the service will be the ip address, logical volume, filesystem, and an init script for the Caché database. This will also be a good time to continue the installation, configuration and testing of the database. After we have manually tested that each of the resources work on its own, then each resource can be added to the cluster to become a resource and eventually a managed service.

For the IP address we have allocated 172.31.224.2 from the subnet to be the shared IP for the cluster. The volume and filesystem were created and tested in the above steps. The init scripts were copied in to place on each cluster node and are located at /etc/init.d/cache, and an example of the scripts are referenced below.

Now we can focus our attention on configuring the database, and a walk through of the major steps follow.

1. Activate cluster logical volume on one node.

```
[root@db1 ~]# lvchange -an /dev/vgcache/lvcache
```

2. Create a mount-point for the volume to be mounted.

```
[root@db1 ~]# mkdir /cache/
```

3. Mount volume at mount-point.

```
[root@db1 ~]# mount /dev/vgcache/lvcache /cache/
```

4. Change to temporary directory holding Cache software.

```
[root@db1 ~]# cd /tmp/cache
```



5. Run the cinstall script and choose the mount-point created above and other appropriate options.

```
[root@db1 cache]# ls
2011.1.2.701.0su_lnxrh5x64.tar.gz  copyright.pdf  docs      LICENSE  tools
cinstall                          cplatname    kitlist   NOTICE
cinstall_client                   dist         lgpl.txt  package
[root@db1 cache]# ./cinstall
```

Your system type is 'Red Hat Enterprise Linux 5 (x64)'.

Currently defined instances:

```
Enter instance name: cdb1
Do you want to create cache instance 'cdb1' <Yes>?
Enter a destination directory for the new instance.
Directory: /cache
```

```
-----
NOTE: Users should not attempt to access Cache while
      the installation is in progress.
-----
```

```
Select installation type.
  1) Development - Install Cache server and all language bindings
  2) Server only - Install Cache server
  3) Custom
Setup type <1>? 2
```

```
Disk blocks required = 1733152
Disk blocks available = 8832456
```

```
Do you want to install Cache Unicode support <No>?
```

```
How restrictive do you want the initial Security settings to be?
"Minimal" is the least restrictive, "Locked Down" is the most secure.
```

```
  1) Minimal
  2) Normal
  3) Locked Down
Initial Security settings <1>? 2
```

```
What user should be the owner of this instance? cache
A Cache account will also be created for user cache.
```

```
Install will create the following Cache accounts for you:
_SYSTEM, Admin, SuperUser, CSPSystem and cache.
Please enter the passwords for these accounts:
Re-enter the password to confirm it:
```

```
What group should be allowed to start and stop
this instance? cache
```

```
Do you want to configure the CSP Gateway to use an existing web server
<No>? yes
```

```
Specify the WebServer type. Choose "None" if you want to configure
```



```
your WebServer manually.
  1) Apache
  2) SunOne
  3) None
WebServer type <3>? 1

Please enter location of Apache configuration file
</etc/httpd/conf/httpd.conf>:

Install detected that Apache httpd server was previously configured for
CSP Gateway. If required, please update httpd server configuration
manually.

Cache did not detect a license key
in directory /cache/mgr.
Do you want to enter a license key <No>?

Please review the installation options:
-----
Instance name: cdb1
Destination directory: /cache
Cache version to install: 2011.1.2.701.0
Installation type: Server
Unicode support: N
Initial Security settings: Normal
User who owns instance: cache
Group allowed to start and stop instance: cache
Effective group for Cache processes: cacheusr
Effective user for Cache SuperServer: cacheusr
SuperServer port: 1972
WebServer port: 57772
JDBC Gateway port: 62972
CSP Gateway: installed into /opt/cspgateway
Client components: none
-----

Do you want to proceed with the installation <Yes>?
...
You can point your browser to http://localhost:57772/csp/sys/UtilHome.csp
to access the system management portal.

Installation completed successfully
```

6. Once the script completes stop cache service with the cache cluster script.

```
[root@db1 cache]# sh /etc/init.d/cache stop
Stopping Cache-HA instance CDB1 on db1
Cache-HA instance CDB1 stopped
```

7. Unmount storage and deactivate logical volume.

```
[root@db1 init.d]# umount /cache
[root@db1 init.d]# lvchange -an /dev/vgcache/lvcache
```

8. Repeat process on other node choosing the same options as the first node.



Now that each of the nodes are configured with the database and scripts tested, the next step in the process will be to add the IP, LVM, FS, and CacheDB script to the cluster as a managed resource. Each of these resources can be created from within the cluster server web interface. Once the resources have been created from within the conga web-ui, the next step will be to create the service called cachedb. The service will provide the coordination for starting and stopping each of the resources in a controlled order as described by the service script in '/usr/share/cluster/service.sh'

Some considerations for the service functioning successfully will be that each node already includes the InterSystems Caché server installed, and configured for the cluster to manage the process. Additionally, the shared storage is configured with the volume group containing a cluster tag, an available virtual ip address dedicated to the cluster service, and an overall quorate working cluster.

While you have the option to create each of the resources during the creation of the service, it is typically more convenient to create each of your resources first. This will provide a library of resources over time to use when building future services. This walk through of creating a service will begin with the creation of four resources. We will use the web conga interface for the task of building the resources.

## Creating the Cluster Service Resources

The first resource that we will create will be the ip address. Here is the list of steps.

- Click on the '*cluster*' tab.
- Click on your cluster name and in the reference case it is '*cachedb1*'
- Click on the '*Resources*' button on the lower left side of the screen.
- Click on the '*Add a Resource*' button on the lower left side of the screen.
- Click on the '*Select a Resource Type*' drop-down button and choose '*IP adress*'.
- Fill out the screen with the information as appropriate to your environment using the following screen shots as a reference and click the '*Submit*' button.

The screenshot shows a web interface for configuring a resource. The page title is "cachedb1" and the subtitle is "Configure 172.31.224.2". The main content area is titled "IP Address Resource Configuration" and contains a form with the following fields: "IP address" with the value "172.31.224.2", "Monitor link" with a checked checkbox, and a "Submit" button. On the left side, there is a navigation menu with "clusters" selected, and under "cachedb1", "Resources" is selected.

The second resource that will be created is the script resource.

- Click on the '*Add a Resource*' button on the lower left side of the screen.
- Click on the '*Select a Resource Type*' drop-down button and choose '*Script*'.



- You will need to provide a name for the script and in the reference we call it '*caché*'. You will also need to provide the path to the script and in the reference it is '*/etc/init.d/caché*' and click the '*Submit*' button.

The screenshot shows the OpenShift web console interface. On the left, there are two navigation menus. The top menu is for 'clusters' with options: 'Cluster List', 'Create a New Cluster', and 'Configure'. The bottom menu is for 'cachedb1' with options: 'Nodes', 'Services', 'Resources', and 'Add a Resource'. The main content area is titled 'cachedb1' and 'Configure cachedb'. It features a 'Script Resource Configuration' section with two input fields: 'Name' containing 'cachedb' and 'Full path to script file' containing '/etc/init.d/cache'. A 'Submit' button is located below the second input field.

The next resource that will be created is the filesystem.

- Click on the '*Add a Resource*' button on the lower left side of the screen.
- Click on the '*Select a Resource Type*' drop-down button and choose '*File system*'.
- Fill out the screen with the appropriate information as appropriate to your environment using the following screen shots as a reference. The items you will need to supply will be the name of the filesystem resource, filesystem, device, mount point, relevant mount options and the force unmount option. In the reference build the the following were used:

*Name:* "cachéfs"

*Filesystem:* "ext3"

*Mount Point:* "/caché"

*Device:* "/dev/vgcache/lvcache"

*Options:* "noatime"

- Once the form is correctly filled out, click the '*Submit*' button.



clusters

Cluster List  
Create a New Cluster  
Configure

cachedb1

Nodes  
Services  
Resources  
Add a Resource  
Configure a Resource  
172.31.224.2  
cachefs  
cachevol  
cachedb

cachedb1

Configure cachefs

**File System Resource Configuration**

Name: cachefs  
File system type: ext3  
Mount point: /cache  
Device: /dev/vgcache/lvcache  
Options: noatime  
File system ID (optional): 25316  
Force unmount:   
Reboot host node if unmount fails:   
Check file system before mounting:

Submit

The final resource that will be created is LVM.

- Click on the 'Add a Resource' button on the lower left side of the screen.
- Click on the 'Select a Resource Type' drop-down button and choose 'LVM'.
- Fill out the screen with the appropriate information as appropriate to your environment using the following screen shots as a reference. The items that you will need to supply will be the name of the resource, volume group and logical group names. In the reference build the following were used: Name *cachevol* Volume Group *vgcache* Logical Volume *lvcache*. Once the form is correctly filled out, click the 'Submit' button.

clusters

Cluster List  
Create a New Cluster  
Configure

cachedb1

Nodes  
Services  
Resources  
Add a Resource  
Configure a Resource

cachedb1

Configure cachevol

**LVM Resource Configuration**

Name: cachevol  
Volume Group Name: vgcache  
Logical Volume Name: lvcache  
Fence the node if it is unable to clean up LVM tags:

Submit

**Building the service.** Now that each of the resources are created, you should now be able to move to building the actual service from the library of available resources.

- Click on the 'Services' button on the lower left side of the screen.
- Click on the 'Add a Service' button.
- Add a Name for the Cluster, and in the reference build it is called 'cache**db1**'
- Choose the 'Automatically start this service' option.
- Choose a preconfigured Failover Domain, and in the reference build it is called



*'ECDomain'*.

- Choose the recovery policy *'Relocate'*
- Now click on the *'Add a resource to this service'* button.
- After choosing *'Add a resource to this service'* then click on the drop-down button *'Use an existing global resource'* and choose the preconfigured resource. In the case of this reference build it is *172.31.224.2 (IP Address)*
- Now click on the *'Add a resource to this service'* button.
- After choosing *'Add a resource to this service'* then click on the drop-down button *'Use an existing global resource'* and choose the preconfigured resource. In the case of this reference build it is *'cachedb (Script)'*.
- Now click on the *'Add a resource to this service'* button.
- After choosing *'Add a resource to this service'* then click on the drop-down button *'Use an existing global resource'* and choose the preconfigured resource. In the case of this reference build it is *'cachefs (File System)'*.
- Now click on the *'Add a resource to this service'* button.
- After choosing *'Add a resource to this service'* then click on the drop-down button *'Use an existing global resource'* and choose the preconfigured resource. In the case of this reference build it is *'cachevol (Logical Volume Management)'*.
- Now that the service definition is complete the final step to creating a service is to click on the *'Submit'* button at the bottom.

## List of preconfigured resources that were used for the service called cachedb in the cluster called cachedb1.

Resource Name	Type
172.31.224.2	IP Address
cache	Script
cachefs	File System
cachevol	Logical Volume Management



## Service Composition

### IP Address Resource Configuration

IP address

Monitor link

This resource is an independent subtree

This resource is non-critical

Maximum number of restart failures before giving up  
(applies only for non-critical resources)

Restart expire time  
(applies only for non-critical resources)

### File System Resource Configuration

Name

File system type

Mount point

Device

Options

File system ID (optional)

Force unmount

Reboot host node if unmount fails

Check file system before mounting

This resource is an independent subtree

This resource is non-critical

Maximum number of restart failures before giving up  
(applies only for non-critical resources)

Restart expire time  
(applies only for non-critical resources)



### LVM Resource Configuration

Name	<input type="text" value="cachevol"/>
Volume Group Name	<input type="text" value="vgcache"/>
Logical Volume Name	<input type="text" value="lvcache"/>
Fence the node if it is unable to clean up LVM tags	<input type="checkbox"/>
This resource is an independent subtree	<input type="checkbox"/>
This resource is non-critical	<input type="checkbox"/>
Maximum number of restart failures before giving up (applies only for non-critical resources)	<input type="text"/>
Restart expire time (applies only for non-critical resources)	<input type="text"/>
<input type="button" value="Add a child"/> <input type="button" value="Delete this resource"/>	

### Script Resource Configuration

Name	<input type="text" value="cachedb"/>
Full path to script file	<input type="text" value="/etc/init.d/cache"/>
This resource is an independent subtree	<input type="checkbox"/>
This resource is non-critical	<input type="checkbox"/>
Maximum number of restart failures before giving up (applies only for non-critical resources)	<input type="text"/>
Restart expire time (applies only for non-critical resources)	<input type="text"/>
<input type="button" value="Add a child"/> <input type="button" value="Delete this resource"/>	

Automatically start this service	<input checked="" type="checkbox"/>
Enable NFS lock workarounds	<input type="checkbox"/>
Run exclusive	<input type="checkbox"/>
Failover Domain	<input type="text" value="ECDomain"/>
Recovery policy	<input type="text" value="Relocate"/>
Maximum number of restart failures before relocating	<input type="text" value="0"/>
Length of time in seconds after which to forget a restart	<input type="text" value="0"/>
<input type="button" value="Add a resource to this service"/> <input type="button" value="Save changes"/>	



## Enabling the Service

Once each of the resources are configured and used to create the service cachedb, then the next step will be to enable the service. This can be done through the conga-ui. Once the service is active, you can check its status either through the web interface or by using the commands `clustat` and `cman_tool status` on each node. You may also want to check that logical volume is active on only one node, the filesystem is mounted, and that you can reach the web interface of the database.

```
[root@db1 ~]# clustat
Cluster Status for cachedb1 @ Wed Nov 30 18:15:04 2011
Member Status: Quorate

Member Name                ID    Status
-----
priv2.clunet.redhat.com    1    Online, rgmanager
priv1.clunet.redhat.com    2    Online, Local, rgmanager
/dev/vdb1                  0    Offline, Quorum Disk

Service Name                Owner (Last)                State
-----
service:cachedb            priv2.clunet.redhat.com    started

[root@db1 ~]# cman_tool status
Version: 6.2.0
Config Version: 79
Cluster Name: cachedb1
Cluster Id: 25665
Cluster Member: Yes
Cluster Generation: 640
Membership state: Cluster-Member
Nodes: 2
Expected votes: 3
Total votes: 2
Node votes: 1
Quorum: 2
Active subsystems: 10
Flags: Dirty
Ports Bound: 0 11 177
Node name: priv1.clunet.redhat.com
Node ID: 2
Multicast addresses: 239.192.100.165
Node addresses: 192.168.0.61

[root@db2 ~]# clustat
Cluster Status for cachedb1 @ Wed Nov 30 23:16:02 2011
Member Status: Quorate

Member Name                ID    Status
-----
priv2.clunet.redhat.com    1    Online, Local, rgmanager
priv1.clunet.redhat.com    2    Online, rgmanager
/dev/vdb1                  0    Online, Quorum Disk

Service Name                Owner (Last)                State
-----
```



```
service:dbserv          priv2.clunet.redhat.com    started

[root@db2 ~]# cman_tool status
Version: 6.2.0
Config Version: 79
Cluster Name: cachedb1
Cluster Id: 25665
Cluster Member: Yes
Cluster Generation: 640
Membership state: Cluster-Member
Nodes: 2
Expected votes: 3
Quorum device votes: 1
Total votes: 3
Node votes: 1
Quorum: 2
Active subsystems: 10
Flags: Dirty
Ports Bound: 0 11 177
Node name: priv2.clunet.redhat.com
Node ID: 1
Multicast addresses: 239.192.100.165
Node addresses: 192.168.0.62
```

Here is an example of the cluster configuration file. This file will be on each node that is a part of the cluster and it is important that each cluster node is running same version of the cluster file.

```
[root@db1 ~]# cat /etc/cluster/cluster.conf
<?xml version="1.0"?>
<cluster alias="cachedb1" config_version="79" name="cachedb1">
  <fence_daemon clean_start="0" post_fail_delay="0" post_join_delay="3"/>
  <clusternodes>
    <clusternode name="priv2.clunet.redhat.com" nodeid="1" votes="1">
      <fence>
        <method name="1">
          <device domain="db2" name="vfence"/>
        </method>
        <method name="2">
          <device name="scsifence" node="db1"/>
        </method>
      </fence>
    </clusternode>
    <clusternode name="priv1.clunet.redhat.com" nodeid="2" votes="1">
      <fence>
        <method name="1">
          <device domain="db1" name="vfence"/>
        </method>
        <method name="2">
          <device name="scsifence" node="db1"/>
        </method>
      </fence>
    </clusternode>
  </clusternodes>
  <cman expected_votes="3" quorum_dev_poll="59000"/>
</fencedevices>
```



```
<fencedevice agent="fence_xvm" name="vfence"/>
<fencedevice agent="fence_scsi" name="scsifence"/>
</fencedevices>
<rm>
  <failoverdomains>
    <failoverdomain name="ECDomain" nofailback="1" ordered="1"
restricted="1">
      <failoverdomainnode name="priv2.clunet.redhat.com"
priority="1"/>
      <failoverdomainnode name="priv1.clunet.redhat.com"
priority="2"/>
    </failoverdomain>
  </failoverdomains>
  <resources>
    <ip address="172.31.224.2" monitor_link="1"/>
    <script file="/etc/init.d/cache" name="cachedb"/>
    <fs device="/dev/vgcache/lvcache" force_fsck="0"
force_unmount="1" fsid="45160" fstype="ext3" mountpoint="/cache" name="cachefs"
options="noatime" self_fence="0"/>
    <lvm lv_name="lvcache" name="cachevol" vg_name="vgcache"/>
  </resources>
  <service autostart="1" domain="ECDomain" exclusive="0"
name="cachedb" recovery="relocate">
    <ip ref="172.31.224.2"/>
    <lvm ref="cachevol"/>
    <fs ref="cachefs"/>
    <script ref="cachedb"/>
  </service>
</rm>
<totem consensus="4800" join="60" token="59000"
token_retransmits_before_loss_const="20"/>
<quorumd interval="4" label="qdisk" log_facility="local4" loglevel="7"
master_wins="1" reboot="0" tko="8" tko_up="2" votes="1"/>
<fence_xvmd/>
</cluster>
```

The final steps will be to navigate to the virtual shared resolvable hostname that we configured as one of steps to creating the cluster. For this reference architecture the link is <http://cache.privnet.redhat.com:57772/csp/sys/UtilHome.csp>.



## 8 Conclusion

In conclusion this document is a reference for customers considering to deploy a clustered Caché DB on Red Hat Enterprise Linux 5. The document covers a number of topics from the initial requirements through the software installation and configuration and continues on through building the cluster. While the expectation of this guide is to provide a complete reference, it is highly recommended to review the additional documentation suggestions in the next section.



## 9 Suggested Reading and References

The following is a listing of recommended reading and references that should provide additional reference and value when working with installation, clustering and tuning of systems.

- [Cluster Administration Guide](#)
- [DM Multipathing Guide](#)
- [Clustered Logical Volume Guide](#)