



Cluster Installation Using SNA HAO for RHEL on System z

Document ID: HAOZ-001

Copyright © 2013,2014 Sine Nomine Associates

All rights reserved. No part of the contents of this material may be reproduced or transmitted in any form or by any means without the written permission of Sine Nomine Associates.

iii

Table of Contents

1	INTRODUCTION	1-1
1.1	Purpose of this Document.....	1-1
1.2	Conventions Used in This Document	1-1
1.3	What Do You Need to Use This Document?.....	1-1
1.3.1	System Resources.....	1-1
1.3.2	Software	1-1
1.3.3	Documentation and References.....	1-2
1.3.4	Access Authorities and Passwords	1-2
1.3.5	Skills.....	1-2
1.4	If You Have Questions or Comments	1-2
2	CLUSTER REFERENCE ARCHITECTURE	2-1
2.1	General Overview of Clustering.....	2-1
2.1.1	Goals of Clustering	2-1
2.1.2	Components	2-1
2.1.3	Configuration.....	2-4
2.2	Cluster Reference Architecture Using HAO	2-1
2.2.1	Components in Example HAO Implementation	2-1
2.2.2	Component Implementation.....	2-3
2.2.3	Networking Implementation	2-3
2.3	Notes.....	2-4
2.3.1	z/VM vs LPAR Implementation.....	2-4
2.3.2	Directory Management Applications Recommendation.....	2-4
3	OBTAINING THE CODE.....	3-1
3.1	Your Userid and Password	3-1
3.2	Accessing the Code	3-1
3.3	Firewall Rules or Other Access Issues	3-1
3.4	If You Have Trouble Obtaining the Code	3-1
3.5	Support.....	3-1

4 INSTALLATION OF INFRASTRUCTURE COMPONENTS FOR A SAMPLE 2 NODE CLUSTER..... 4-1

4.1 Overview	4-1
4.1.1 Introduction	4-1
4.1.2 Task List	4-1
4.2 Creating the z/VM Userid for Shared Cluster Disks.....	4-2
4.2.1 Task: Create User COMMON	4-2
4.3 Creating the z/VM Userids for the Cluster Userids	4-2
4.3.1 Task: Create the source for PROFILE EXEC	4-2
4.3.2 Task: Create User CMANAGER	4-2
4.3.3 Task: Prepare 'A' Disk of CMANAGER.....	4-3
4.3.4 Task: Create User CNODE1.....	4-3
4.3.5 Task: Prepare 'A' Disk of CNODE1	4-4
4.3.6 Task: Create User CNODE2.....	4-4
4.3.7 Task: Prepare 'A' Disk of CNODE2	4-5
4.4 Enabling the z/VM SMAPI Server	4-5
4.4.1 Task: Log On As a Privileged User.....	4-5
4.4.2 Task: Access z/VM SMAPI Configuration Minidisk/SFS Directory	4-5
4.4.3 Task: Edit VSMWORK1 AUTHLIST	4-5
4.4.4 Task: Test VSMWORK1 (optional)	4-5
4.5 Installing RHEL for System z on CMANAGER, CNODE1 and CNODE2	4-5
4.5.1 Task: Run "yum update" on CMANAGER	4-6
4.5.2 Task: Run "yum update" on CNODE1.....	4-6
4.5.3 Task: Run "yum update" on CNODE2.....	4-6
4.6 Installing the SNA Packages RPM	4-6
4.7 Configuring and Enabling the SNA HAO Repository on Cluster Nodes.....	4-6
4.8 Disable Network Manager on Cluster Nodes	4-7
4.9 Installing the GFS2 Kernel Module	4-7
4.9.1 Task: Install GFS2 Kernel Module on CNODE1 and CNODE2	4-7
4.10 Task: Enabling SELinux Policy for Clustering [SELinux Only]	4-7
4.11 Task: Enabling Network Ports on CMANAGER [Firewall Only].....	4-7
4.12 Enabling Network Ports on CNODE1 and CNODE2 [Firewall Only]	4-8

4.13	Installing the SNA HAO luci code on CMANAGER.....	4-9
4.14	Configure Administrator User on luci	4-9
4.15	Installing the SNA HAO ricci code on CNODE1 and CNODE2.....	4-12
5	CREATING THE CLUSTER.....	5-14
5.1.1	Task: Create Cluster.....	5-14
5.2	Add CNODE1 and CNODE2 to Cluster	5-14
5.3	Task: Add a Fence Device	5-17
5.4	Task: Add Fence Methods to CNODE1 and CNODE2	5-18
5.5	Task: Additional Considerations for SSI Environments.....	5-20
5.6	Task: Testing Fencing	5-20
6	CONFIGURING CLUSTER RESOURCES AND SERVICES	6-1
6.1	Formatting Shared GFS2 Disk	6-1
6.1.1	Task: Activate Shared GFS2 Disk on CNODE1 and CNODE2	6-1
6.1.2	Task: Format and Partition Shared Cluster Disks on CNODE1.....	6-1
6.2	Setting Up a GFS2 Filesystem	6-1
6.2.1	Task: Create Physical Volume.....	6-1
6.2.2	Task: Create Cluster Volume Group	6-1
6.2.3	Task: Make Volume Group Visible from CNODE2	6-2
6.2.4	Task: Create Logical Volume	6-2
6.2.5	Task: Create GFS2 Filesystem	6-2
6.3	Creating HA Web Server on CNODE1 and CNODE2	6-2
6.3.1	Task: Install Apache	6-2
6.4	Create Cluster Resources	6-2
6.4.1	Task: Create Web Server Resource	6-2
6.4.2	Task: Create GFS2 Resource	6-3
6.4.3	Task: Create IP Address Resource	6-4
6.5	Task: Create a Cluster Failover Domain	6-5
6.6	Create a Cluster Service Group.....	6-5
6.6.1	Task: Create the Service Group	6-6
6.6.2	Task: Add GFS2 Resource	6-6

6.6.3	Task: Add IP Address Resource	6-7
6.6.4	Task: Add Web Service Resource	6-7
6.7	Task: Starting the Service	6-8
6.8	Task: Create Home Page	6-8
6.9	Task: Point Browser at HA IP Address	6-9
6.10	Task: Failover Web Service to CNODE2.....	6-1
6.11	Task: Point Browser at HA IP Address.....	6-3
7	COMMON VARIATIONS TO THIS DESIGN.....	7-1
7.1	Adding a Quorum Disk	7-1
7.1.1	Task: Add Volumes to User COMMON	7-1
7.1.2	Task: Activate Volume on CNODE1	7-1
7.1.3	Task: Format and Partition New Volume on NODE1.....	7-2
7.1.4	Task: Make the Quorum Disk File System	7-2
7.1.5	Task: Creating Quorum Disk [Optional].....	7-2
7.1.6	Task: Activate Volume on CNODE2	7-2
7.1.7	Task: Define Quorum Disk on CMANAGER.....	7-3
7.2	Where to Find Out More Information	7-4
8	Z/VM FENCE DEVICES	8-1
8.1	fence_zvm	8-1
8.2	fence_zvmip	8-2
9	OTHER FILES.....	9-1
9.1	Virtual Machine A Disks	9-1
9.1.1	SWAPGEN	9-1
9.1.2	PROFILE EXEC	9-1
9.2	SELinux Policy	9-1
9.3	SMAPI Test – RECYCLE EXEC	9-3
10	ADDITIONAL RESOURCES.....	10-7
10.1	HAO Documentation.....	10-7

10.2	IBM z/VM Documentation	10-7
10.3	Red Hat Documentation.....	10-7
10.4	Other Documentation	10-7

List of Tables

Table 1: HAO Software Components.....	2-2
Table 2: Disk Storage and Virtual Address Specifications for Example Cluster	2-3
Table 3: HAO Package Contents	4-6
Table 4: luci iptables requirements	4-8
Table 5: Cluster node iptables requirements	4-8
Table 6: fence_zvm parameters	8-1
Table 7: fence_zvmip parameters	8-2

List of Figures

Figure 1: Example of XML-based Cluster Definition File (created by luci)	2-5
Figure 2: luci login screen.....	4-10
Figure 3: luci Action menu	4-11
Figure 4: luci login screen.....	4-11
Figure 5: luci Add clusteradm user	4-11
Figure 6: luci Define clusteradm user privileges	4-12
Figure 7: luci Create cluster.....	5-14
Figure 8: luci Add nodes to cluster.....	5-14
Figure 9: luci Cluster creation progress screen	5-15
Figure 10: luci Cluster nodes added	5-16
Figure 11: luci Add fence device	5-17
Figure 12: luci Add fence instance	5-18
Figure 13: luci Add fence methods to nodes.....	5-18
Figure 14: luci Choosing a name for fence method instance	5-18

Figure 15: luci Adding an instance of fence method	5-19
Figure 16: luci Choosing an existing fence device	5-19
Figure 17: luci Providing parameters to fence instance	5-19
Figure 18: luci Fence definition completed	5-20
Figure 19: Testing fencing	5-20
Figure 20: Testing fencing – node log contents	5-21
Figure 21: Testing fencing – z/VM log contents	5-22
Figure 22: luci Adding Apache Resource	6-2
Figure 23: luci Adding GFS2 Resource	6-3
Figure 24: luci Adding IP Address Resource	6-4
Figure 25: luci Creating a failover domain	6-5
Figure 26: luci Creating a service group	6-6
Figure 27: luci Adding GFS2 resource to the service group	6-6
Figure 28: luci Adding IP resource to the service group	6-7
Figure 29: luci Adding Web Service resource to the service group	6-7
Figure 30: luci Starting the service using Start link	6-8
Figure 31: luci Starting the service on a specific node	6-8
Figure 32: luci Result of starting the service	6-8
Figure 33: Accessing service via Browser	6-9
Figure 34: Testing Failover	6-3
Figure 35: luci Defining a Quorum Disk	7-3
Figure 36: luci Quorum Disk Created	7-4
Figure 37: Sample PROFILE EXEC	9-1
Figure 38: SELinux Definitions	9-2
Figure 39: SMAPI Test - RECYCLE EXEC	9-3

1 Introduction

1.1 Purpose of this Document

The document serves as an installation guide and architectural reference for the Sine Nomine High-Availability Option for RHEL on System z. The document describes the overall architecture of clustering systems, the HAO installation process, and provides an example of building a two-node cluster administered by the “luci” management tool is described in detail.

Clustering is not something to be undertaken lightly. The goals of why you are implementing clustering need to be well understood. Cluster configuration, administration, and operation require a high level of expertise beyond that needed for normal Linux systems.

If you are familiar with the High Availability Option on x86/x86_64 platforms then nearly everything will be the same for HAO on System z. The major differences boil down to:

1. Different fence devices
2. Definition of shared resources (e.g. GFS2 volumes)

1.2 Conventions Used in This Document

1.3 What Do You Need to Use This Document?

To make effective use of this document, we assume you have the following resources, skills and documents available for your install and to create the sample cluster.

1.3.1 System Resources

- A System z processor capable of running z/VM 5.4 or higher. The example cluster implementation in this paper was implemented on a z196.

1.3.2 Software

- z/VM release 5.4 or higher. The example cluster in this document was implemented on z/VM 6.2 without SSL.
- Red Hat Enterprise Linux (RHEL) 6.x for System z. The example cluster in this document was implemented using RHEL version 6.4
- A z/VM directory manager such as CA VM:Secure or IBM DIRMAINT (see topic 2.3.2 “Directory Management Applications Recommendation” on page 2-4).
- The SNA HAO for RHEL on System z “yum” repository. Obtaining access to this repository is described in section 3 “Obtaining the Code” on page 3-1 of this document.

1.3.3 Documentation and References

- Access to the SNA HAO documentation (this document).
- Access to the IBM documentation referenced in section 10.2 “IBM z/VM Documentation” on page 10-7.

1.3.4 Access Authorities and Passwords

When you sign up for the trial or full support you will be provided with access information including user/password details that will be used to configure the yum repository information.

1.3.5 Skills

Sound Linux system administration skills are essential. Knowledge of clustering is recommended but the information provided within this document should be sufficient to get you started.

1.4 If You Have Questions or Comments

Please contact support@sinenomine.net.

2 Cluster Reference Architecture

2.1 General Overview of Clustering

In this section the goals, components and configuration of clustering are described.

2.1.1 Goals of Clustering

The following extract from Wikipedia concisely defines HA clustering:

“High-availability clusters are groups of computers that support server applications that can be reliably utilized with a minimum of down-time. They operate by harnessing redundant computers in groups or clusters that provide continued service when system components fail. Without clustering, if a server running a particular application crashes, the application will be unavailable until the crashed server is fixed. HA clustering remedies this situation by detecting hardware/software faults, and immediately restarting the application on another system without requiring administrative intervention, a process known as failover. As part of this process, clustering software may configure the node before starting the application on it. For example, appropriate filesystems may need to be imported and mounted, network hardware may have to be configured, and some supporting applications may need to be running as well.

“HA clusters are often used for critical databases, file sharing on a network, business applications, and customer services such as electronic commerce websites.

“HA cluster implementations attempt to build redundancy into a cluster to eliminate single points of failure, including multiple network connections and data storage which is redundantly connected via storage area networks.

“HA clusters usually use a heartbeat private network connection which is used to monitor the health and status of each node in the cluster.” [7]

2.1.2 Components

The major software components of the HA option includes:

- Cluster infrastructure — Provides fundamental functions for nodes to work together as a cluster
- Configuration-file management, membership management, lock management, and fencing
- High availability Service Management — Provides failover of services from one cluster node to another in case a node becomes inoperative
- Cluster administration tools — Configuration and management tools for setting up, configuring, and managing the High Availability Implementation
- Red Hat GFS2 (Global File System 2) — Provides a cluster file system for use with the High Availability Add-On. GFS2 allows multiple nodes to share storage at a block level as if the storage were connected locally to each cluster node
- Cluster Logical Volume Manager (CLVM) — Provides volume management of cluster storage
- Optional load balancing component to enable even allocation of work.

These functions are implemented via the following services:

Table 1: HAO Software Components

Service	Description
cman	<p>Cluster management manages cluster quorum and cluster membership. CMAN (an abbreviation for cluster manager) performs cluster management in the High Availability Add-On for Red Hat Enterprise Linux. CMAN is a distributed cluster manager and runs in each cluster node; cluster management is distributed across all nodes in the cluster.</p> <p>CMAN keeps track of membership by monitoring messages from other cluster nodes. When cluster membership changes, the cluster manager notifies the other infrastructure components, which then take appropriate action. If a cluster node does not transmit a message within a prescribed amount of time, the cluster manager removes the node from the cluster and communicates to other cluster infrastructure components that the node is not a member. Other cluster infrastructure components determine what actions to take upon notification that node is no longer a cluster member. For example, Fencing would fence the node that is no longer a member.</p> <p>CMAN keeps track of cluster quorum by monitoring the count of cluster nodes. If more than half the nodes are active, the cluster has quorum. If half the nodes (or fewer) are active, the cluster does not have quorum, and all cluster activity is stopped. Cluster quorum prevents the occurrence of a "split-brain" condition — a condition where two instances of the same cluster are running. A split-brain condition would allow each cluster instance to access cluster resources without knowledge of the other cluster instance, resulting in corrupted cluster integrity. [1]</p>
rgmanager	<p>Resource manager - provides high availability of critical server applications in the event of planned or unplanned system downtime. rgmanager handles management of user-defined cluster services (also known as resource groups). This includes handling of user requests including service start, service disable, service relocate, and service restart. The service manager daemon also handles restarting and relocating services in the event of failures.</p>
ricci	<p>Cluster and storage management and configuration daemon. The ricci daemon, dispatches incoming messages to underlying management modules. It consists of the listener daemon (dispatcher), as well as reboot, rpm, storage, service, virtual machine, and log management modules.</p>
modclusterd	<p>The modclusterd service provides an abstraction of cluster status used by luci and by the Simple Network Management (SNMP) and Common Information Model (CIM) modules of clustermon.</p>

Service	Description
clvmd	Cluster-aware logical volume manager. clvmd is the daemon that distributes LVM metadata updates around a cluster. It must be running on all nodes in the cluster and will give an error if a node in the cluster does not have this daemon running.
qdiskd	The Quorum Disk daemon talks to CMAN and provides a mechanism for determining node-fitness in a cluster environment.
fenced	The fencing daemon, fenced, fences cluster nodes that have failed. Fencing a node generally means rebooting it or otherwise preventing it from writing to storage, e.g. disabling its port on a SAN switch. Fencing involves interacting with a hardware device, e.g. network power switch, SAN switch, and storage array. Different "fencing agents" are run by fenced to interact with various hardware devices.
dml_controld	The distributed lock manager (dlm) lives in the kernel, and the cluster infrastructure (corosync membership and group management) lives in user space. The dlm in the kernel needs to adjust/recover for certain cluster events. It's the job of dml_controld to receive these events and reconfigure the kernel dlm as needed. dml_controld controls and configures the dlm through sysfs and configfs files that are considered dlm-internal interfaces.
luci	<p>luci makes it easy to handle most of common tasks connected with cluster/HA suite management. Such administrative tasks luci provides include:</p> <ul style="list-style-type: none"> Inspect various aspects of managed clusters Manipulate: <ul style="list-style-type: none"> Whole clusters: create a new cluster of specified nodes, add existing to/remove selected from the "awareness of luci" Cluster nodes: add, remove, reboot and configure Services within cluster: add, delete, start, restart, disable and configure properties Resources within cluster: add, delete and configure resource-specific properties Failover domains within cluster: add, delete and configure properties Global fence device instances within cluster: add, delete and configure global properties specific to particular fence device instance Configure some properties of the whole cluster: <ul style="list-style-type: none"> General properties Fence daemon properties Network configuration Quorum disk configuration <p>It is quite common to run luci on another machine than are the cluster's nodes running ricci daemon in order to enable luci to manage the cluster.</p>
corosync	Corosync provides clustering infrastructure such as membership, messaging and quorum. It is a group communication system with additional features for implementing

Service	Description
	<p>high availability within applications. Loadable modules, called service engines, are loaded into the Corosync Cluster Engine and use the services provided by the Corosync Service Engine internal API.</p> <p>The services provided by the Corosync Service Engine internal API are:</p> <ul style="list-style-type: none"> • An implementation of the Totem Single Ring Ordering and Membership protocol providing the Extended Virtual Synchrony model for messaging and membership. • The coroipec high performance shared memory IPC system. • An object database that implements the in memory database model. • Systems to route IPC and Totem messages to the correct service engines. <p>Additionally Corosync provides several default service engines:</p> <ul style="list-style-type: none"> • cpg - Closed Process Group • sam - Simple Availability Manager • confdb - Configuration and Statistics database • quorum - Provides notifications of gain or loss of quorum
gfs2.ko	GFS2 file system driver kernel module

2.1.3 Configuration

Clusters are configured by the creation and modification of the cluster configuration file `/etc/cluster/cluster.conf`, which lives on each node of the cluster. This file can be maintained by hand but this is subject to all the problems associated with manual file administration. Typically, the `luci` tool (formerly known as “conga”) is used to create, maintain and distribute this file.

A typical configuration file looks like the following:

2.2 Cluster Reference Architecture Using HAO

In this section a reference architecture using the High Availability Option is described.

2.2.1 Components in Example HAO Implementation

A HAO implementation is composed of software, hypervisor, and virtual machines.

2.2.1.1 Software Components

- Hypervisor (z/VM 6.2)
- Red Hat Enterprise Linux 6.4
- HAO for RHEL on System z
- GFS2 Kernel Module for HAO for RHEL on System z

2.2.1.2 z/VM 6.2 Hypervisor

z/VM provides a highly flexible test and production environment on the IBM System z platform. The z/VM implementation of IBM virtualization technology provides the capability to run full function operating systems such as Linux on IBM System z, z/OS, z/VSE, z/TPF, and z/VM "guests" of z/VM. z/VM supports 64-bit IBM z/Architecture® guests and 31-bit IBM Enterprise Systems Architecture/390 (ESA/390) guests. [2]

Function

The heart of the VM architecture is a control program or hypervisor called VM-CP (usually, and sometimes, ambiguously: VM). It runs on the physical hardware, and creates the virtual machine environment. VM-CP provides full virtualization of the physical machine – including all I/O and other privileged operations. It performs the system's resource-sharing, including device management, dispatching, virtual storage management, and other traditional operating system tasks. Each VM user is provided with a separate virtual machine having its own address space, virtual devices, etc., and which is capable of running any software that could be run on a standalone machine. A given VM mainframe typically runs hundreds or thousands of virtual machine instances. [6]

Software Installed

1. A directory management product such as DIRMAINT or VM:Secure that will be used to define new virtual machines as well as interact with the Systems Management API (SMAPI) used by the cluster fence devices.
2. A working TCP/IP stack providing access via VSWITCH or Guest LANs to the virtual machines.

2.2.1.3 Directory Manager

The use of a directory manager like VM:Direct, VM:Secure or DIRMAINT is highly recommended. In fact I would go as far as to say that it is almost essential.

2.2.1.4 Virtual Machines

COMMON

Function

Owner of shared resources: i.e. gfs2 storage, quorum disk.

Software Installed

None. This virtual machine is never running but is purely a resource owner.

CMANAGER

Function

Cluster manager running luci administrative tool. Separates cluster configuration from cluster operation.

Software Installed

- RHEL 6.4
- luci

CNODE1

Function

A node of the cluster.

Software Installed

- RHEL 6.4
- ricci

Note: the cluster configuration process using luci will cause software to be installed on this node automatically.

CNODE2

Function

A node of the cluster.

Software Installed

- RHEL 6.4

- ricci

Note: the cluster configuration process using luci will cause software to be installed on this node automatically.

2.2.1.5 Storage Needed

Table 2: Disk Storage and Virtual Address Specifications for Example Cluster

Virtual Machine	Description	Address	Size
COMMON	GFS2 – shared production file system(s)	153	3338 cyl
	Quorum disk (optional)	200	20 cyl
CMANAGER	RHEL 6.4 system	150-151	2 x 3338 cyl
	Swap – VDISK	152	200000 blocks
CNODE1	RHEL 6.4 system	150-151	2 x 3338 cyl
	Swap – VDISK	152	200000 blocks
CNODE2	RHEL 6.4 system	150-151	2 x 3338 cyl
	Swap – VDISK	152	200000 blocks

2.2.2 Component Implementation

2.2.2.1 Hypervisor

This document assumes you have a working z/VM 6.2 system (or later) with access to sufficient disk resources as described above. The configuration described in the following sections may be implemented on a single z/VM system or across multiple instances.

2.2.3 Networking Implementation

2.2.3.1 Hypervisor

On the user PMAINT, update the SYSTEM CONFIG file to permanently define a layer-2 virtual switch:

1. #CP CPRELEASE A
2. #CP LINK * CF0 CF0 MR
3. ACC CF0 F
4. Insert the following statement in your SYSTEM CONFIG file on PMAINT's CF0 disk:

```
DEFINE VSWITCH VSWITCH2 RDEV 1340 CONTROLLER DTCVSW2 ETH CONNECT
```

5. REL F
6. #CP LINK * CFO CFO SR
7. #CP CPACCESS PMAINT CFO A

To make it available immediately issue the following command:

```
#CP DEFINE VSWITCH RDEV 1340 CONTROLLER DTCVSW2 ETH CONNECT
```

2.3 Notes

2.3.1 z/VM vs LPAR Implementation

The fencing agent provided by the SNA HAO offering only support a z/VM environment. Future fencing agent(s) may be provided for “bare-iron” operation.

2.3.2 Directory Management Applications Recommendation

The component of z/VM that the fence agents use to perform their power-fence operations is known as SMAPI. SMAPI is designed to work best with a full-function directory management product like DIRMAINT or VMSECURE. It is **strongly** recommended that such a product be installed and operational.

However, it is possible to run SMAPI using a minimally function DIRMAINT system that comes preinstalled with z/VM. The only difference is that for those without a license for DIRMAINT the product will come up in a disabled mode that only provides enough function to enable SMAPI operations. But, just because you can do something doesn't mean you should! Take the time and configure a directory manager, they're cheap, they work and you will avoid problems down the road.

3 Obtaining the Code

3.1 Your Userid and Password

These will be provided at the time you sign up for the offering. These credentials are used to complete the yum repository information (see “4.7 Configuring and Enabling the SNA HAO Repository on Cluster Nodes” 4-6).

3.2 Accessing the Code

You will be provided with the URL of the HAO materials.

3.3 Firewall Rules or Other Access Issues

The HAO fence agent uses either TCP/IP or IUCV to perform the fencing operation. The former requires a port to be opened. In addition, some of the clustering demons also require the use of certain ports. These details may be found at “4.11 Task: Enabling Network Ports on CMANAGER [Firewall Only]” on page 4-7 and “4.12 Enabling Network Ports on CNODE1 and CNODE2 [Firewall Only]” on page 4-8. In addition, if you are using SELinux you will need to read “4.10 Task: Enabling SELinux Policy for Clustering [SELinux Only]” on page 4-7.

3.4 If You Have Trouble Obtaining the Code

Please contact us at support@sinenomine.net.

3.5 Support

Our problem tracking system will be enabled to receive your support requests using the support@sinenomine.net address.

4 Installation of Infrastructure Components for a Sample 2 Node Cluster

4.1 Overview

In this section the steps necessary to prepare the z/VM components are described. This document assumes the use of VM:SECURE or VM:DIRECT.

4.1.1 Introduction

The following tasks are to be performed from a z/VM user that is authorized as a directory manager by VM:SECURE or VM:DIRECT, such as VMANAGER.

4.1.2 Task List

#	Description	<input checked="" type="checkbox"/>
<i>Creating the z/VM Userid for Shared Cluster Disks</i>		
1	Create user COMMON	
<i>Creating the z/VM Userids for Cluster Operation</i>		
1	Prepare the source for PROFILE EXEC	
2	Create user CMANAGER	
3	Prepare A disk of CMANAGER	
4	Create user CNODE1	
5	Prepare A disk of CNODE1	
6	Create user CNODE2	
8	Prepare A disk of CNODE2	
1	Logon as a privileged user	
2	Access z/VM SMAPI Configuration minidisk/SFS directory	
3	Edit VSMWORK1 AUTHLIST	
4	Test VSMWORK1	

4.2 Creating the z/VM Userid for Shared Cluster Disks

The shared disks (GFS2 and quorum) will be owned by a userid that will not be logged on to the system. It simply acts as the owner of the resources to which the cluster nodes will have access.

4.2.1 Task: Create User COMMON

Create a file COMMON DIRECT A containing the following:

```
USER COMMON NOLOG 8K 8K
  MDOPT FORMAT LABEL GFS001
  MDISK 153 3390 * 3338 *
  MINIOPT NOMDC
  MDOPT FORMAT LABEL QUORUM1
  MDISK 200 3390 * 0020 *1
  MINIOPT NOMDC
```

1. See “7.1 Adding a Quorum Disk” on page 7-1

Issue the following command:

```
VMSECURE ADDENTRY COMMON DIRECT A (NOSKEL
```

4.3 Creating the z/VM Userids for the Cluster Userids

4.3.1 Task: Create the source for PROFILE EXEC

Create a file on your ‘A’ disk called CLUSTER PROFILE based on the file shown at “9.1.2 PROFILE EXEC” on page 9-1. This will be used by VMSECURE when creating the virtual machines.

4.3.2 Task: Create User CMANAGER

This virtual machine will be used to administer the cluster: it will not be a node within the cluster. Create a file CMANAGER DIRECT A containing the following, specifying a password that meets your installation standards:

```
USER CMANAGER XXXXXXXX 768M 2G G 64
*SP= CLUSTER PROFILE
*ENROLL 8000 2 VMSYSU (IPL
*AC= 99999999
  ACCOUNT 99999999 LINUX
  MACHINE ESA
  COMMAND SET VSWITCH VSWITCH2 GRANT &USERID
  COMMAND COUPLE C600 TO SYSTEM VSWITCH2
  IUCV ANY
  IPL CMS PARM AUTO CR
  CONSOLE 0009 3215
  SPOOL 00C 2540 READER *
```

```

SPOOL 00D 2540 PUNCH A
SPOOL 00E 1403 A
LINK MAINT 190 190 RR
LINK MAINT 19E 19E RR
NICDEF C600 TYPE QDIO DEVICES 3
MDOPT FORMAT
MDISK 150 3390 * 3338 * M
MDOPT FORMAT
MDISK 151 3390 * 3338 * M

```

Issue the following command:

```
VMSECURE ADDENTRY CMANAGER DIRECT A (NOSKEL
```

VMSECURE will enroll the virtual machine in the VMSYSU Shared File System by executing the command:

```
ENROLL USER CMANAGER VMSYSU (BLOCKS 8000
```

VMSECURE will also create a PROFILE EXEC based on the file CLUSTER PROFILE.

4.3.3 Task: Prepare 'A' Disk of CMANAGER

Acquire and install the SWAPGEN package on the 'A' disk of the user. See "9.1.1 SWAPGEN" on page 9-1.

4.3.4 Task: Create User CNODE1

Create a file CNODE1 DIRECT A containing the following, specifying a password that meets your installation standards:

```

USER CNODE1 XXXXXXXX 768M 2G G
*SP= CLUSTER PROFILE
*ENROLL 8000 2 VMSYSU (IPL
*AC= 99999999
ACCOUNT 99999999 CLUSTER
MACHINE ESA
COMMAND SET VSWITCH VSWITCH2 GRANT &USERID
COMMAND COUPLE C600 TO SYSTEM VSWITCH2
IUCV VSMREQIU
IPL CMS PARM AUTO CR
CONSOLE 0009 3215 T OPERATOR
SPOOL 00C 2540 READER *
SPOOL 00D 2540 PUNCH A
SPOOL 00E 1403 A
LINK MAINT 190 190 RR
LINK MAINT 19E 19E RR
LINK COMMON 153 153 MW
LINK COMMON 200 200 MW1

```



```

NICDEF C600 TYPE QDIO DEVICES 3
MDOPT FORMAT
MDISK 150 3390 * 3338 * M
MDOPT FORMAT
MDISK 151 3390 * 3338 * M

```

1. See “7.1 Adding a Quorum Disk” on page 7-1

Issue the following command:

```
VMSECURE ADDENTRY CMANAGER DIRECT A (NOSKEL
```

4.3.5 Task: Prepare ‘A’ Disk of CNODE1

Acquire and install the SWAPGEN package on the ‘A’ disk of the user. See “9.1.1 SWAPGEN” on page 9-1.

4.3.6 Task: Create User CNODE2

Create a file CNODE2 DIRECT A containing the following, specifying a password that meets your installation standards:

```

USER CNODE2 XXXXXXXX 768M 2G G
*SP= CLUSTER PROFILE
*ENROLL 8000 2 VMSYSU (IPL
*AC= 99999999
ACCOUNT 99999999 CLUSTER
MACHINE ESA
COMMAND SET VSWITCH VSWITCH2 GRANT &USERID
COMMAND COUPLE C600 TO SYSTEM VSWITCH2
IUCV VSMREQIU
IPL CMS PARM AUTO CR
CONSOLE 0009 3215 T OPERATOR
SPOOL 00C 2540 READER *
SPOOL 00D 2540 PUNCH A
SPOOL 00E 1403 A
LINK MAINT 190 190 RR
LINK MAINT 19E 19E RR
LINK COMMON 153 153 MW
LINK COMMON 200 200 MW1
NICDEF C600 TYPE QDIO DEVICES 3
MDOPT FORMAT
MDISK 150 3390 * 3338 * M
MDOPT FORMAT
MDISK 151 3390 * 3338 * M

```

1. See “7.1 Adding a Quorum Disk” on page 7-1

Issue the following command:

```
VMSECURE ADDENTRY CMANAGER DIRECT A (NOSKEL
```

4.3.7 Task: Prepare 'A' Disk of CNODE2

Acquire and install the SWAPGEN package on the 'A' disk of the user. See “9.1.1 SWAPGEN” on page 9-1.

4.4 Enabling the z/VM SMAPI Server

4.4.1 Task: Log On As a Privileged User

```
LOGON MAINT
```

4.4.2 Task: Access z/VM SMAPI Configuration Minidisk/SFS Directory

```
VMLINK .DIR VMSYS:VSMWORK1. (WRITE
```

4.4.3 Task: Edit VSMWORK1 AUTHLIST

To use this agent the z/VM SMAPI service needs to be configured to allow the virtual machine running this agent to connect to it and issue the `image_recycle` operation. This involves adding an entry to the VSMWORK1 AUTHLIST file.

```
XEDIT VSMWORK1 AUTHLIST
```

The new entry should look something similar to this:

Column 1	Column 66	Column 131
↓	↓	↓
CMANAGER	ALL	IMAGE_OPERATIONS

4.4.4 Task: Test VSMWORK1 (optional)

Use the RECYCLE EXEC described at “9.3 SMAPI Test – RECYCLE EXEC” on page 9-3, to test the SMAPI configuration.

4.5 Installing RHEL for System z on CMANAGER, CNODE1 and CNODE2

Follow your installation’s recommended procedures to install Red Hat Enterprise Linux 6.x on the private disk storage for CNODE1 and CNODE2. If your installation has no recommended procedures, the default instructions for installing RHEL are located at https://access.redhat.com/site/documentation/en-US/Red_Hat_Enterprise_Linux/6/pdf/Installation_Guide/Red_Hat_Enterprise_Linux-6-Installation_Guide-en-US.pdf

4.5.1 Task: Run “yum update” on CMANAGER**4.5.2 Task: Run “yum update” on CNODE1****4.5.3 Task: Run “yum update” on CNODE2****4.6 Installing the SNA Packages RPM**

```
yum install
https://USERID:PASSWORD@files.sinenomine.net/hao/cluster/noarch/sna-
packages-1.0-0.noarch.rpm
```

Use the USERID/PASSWORD provided to you for the HA trial in the above URL.

4.7 Configuring and Enabling the SNA HAO Repository on Cluster Nodes

The download materials include the following important files and directories.

Table 3: HAO Package Contents

2.6.32-358	Contains the gfs2 kernel module for 2.6.32-358 kernel
2.6.32-358.0.1	Contains the gfs2 kernel module for 2.6.32-358.0.1
2.6.32-358.2.1	Contains the gfs2 kernel module for 2.6.32-358.2.1
2.6.32-358.23.2	Contains the gfs2 kernel module for 2.6.32-358.23.2
2.6.32.431.11.2	Contains the gfs2 kernel module for 2.6.32-431.11.2
2.6.32.431.17.1	Contains the gfs2 kernel module for 2.6.32-431.17.1
noarch	Architecture independent RPMs
s390x	System z platform specific RPMs
snahao.pp	SELinux policy

Install the sna-packages RPM first. This will create a file in /etc/yum.repos.d directory. The file should contain an entry similar to this:

```
[sna]
name=SNA Public Packages
mirrorlist=http://mirrors.sinenomine.net/sna?releasever=1&arch=$basearc
h&repo=public
baseurl=
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-SNA
```

```
enabled=1
```

```
[sna-ha]
name=SNA High Availability Packages
baseurl=https://USERID:PASSWORD@files.sinenomine.net/hao/cluster
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-SNA
enabled=0
```

Use the USERID/PASSWORD provided to you for the HA trial to update the `baseurl` entry above. Also, you should change `enabled=0` to `enabled=1` for this entry and set `enabled=0` for the `[sna]` entry. You may also install the repository on a local server and modify the `baseurl` settings as you please.

You should now be able to install the necessary software on each of the nodes.

4.8 Disable Network Manager on Cluster Nodes

```
chkconfig --levels 345 NetworkManager off
service NetworkManager stop
```

4.9 Installing the GFS2 Kernel Module

If your kernel level is different to those in the above table, a new kernel module will need to be built. The source RPM is supplied but we can do it for you if it is required.

4.9.1 Task: Install GFS2 Kernel Module on CNODE1 and CNODE2

In this configuration, we are running kernel 2.6.32-358.2.1 so we install the `gfs2` kernel module built for that kernel level. (See “Table 3: HAO Package Contents” on page 4-6.)

```
yum install
https://USERID:PASSWORD@files.sinenomine.net/hao/cluster/2.6.32-
358.2.1/gfs2-kmod-2.0-1.el6.s390x.rpm
depmod -a
modprobe gfs2
```

4.10 Task: Enabling SELinux Policy for Clustering [SELinux Only]

If SELinux is enabled on the hosts, then an additional policy needs to be installed:

```
wget https://USERID:PASSWORD@files.sinenomine.net/hao/cluster/snahao.pp
semodule -i snahao.pp
```

4.11 Task: Enabling Network Ports on CMANAGER [Firewall Only]

If the firewall service is enabled on the host, then the following ports need to be enabled on the cluster nodes:

Table 4: luci iptables requirements

Ports	Type	Description
8084	TCP	luci (Conga user interface server)

In our test environment we are isolated for the outside world so we did not specify source or destination addresses in the rules. Your rules may need to reflect your own network constraints.

For luci (Conga User Interface server):

```
iptables -I INPUT -m state --state NEW -p tcp --dport 8084 -j ACCEPT
```

```
iptables -I OUTPUT -m state --state NEW -p tcp --dport 8084 -j ACCEPT
```

For igmp (Internet Group Management Protocol):

```
iptables -I INPUT -p igmp -j ACCEPT
```

After executing these commands, run the following command to save the current configuration for the changes to be persistent during reboot.

```
service iptables save ; service iptables restart
```

4.12 Enabling Network Ports on CNODE1 and CNODE2 [Firewall Only]

If the firewall service is enabled on the nodes, then the following ports need to be enabled on the cluster nodes:

Table 5: Cluster node iptables requirements

Ports	Type	Description
5404, 5405	UDP	corosync/cman (Cluster Manager)
11111	TCP	ricci (propagates updated cluster information)
21064	TCP	dlm (Distributed Lock Manager)
16851	TCP	modclusterd
44444	TCP	fenced

In our test environment we are isolated for the outside world so we did not specify source or destination addresses in the rules. Your rules may need to reflect your own network constraints.

For cman (Cluster Manager), use the following filtering:

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

```
iptables -I INPUT -m addrtype --dst-type MULTICAST -m state --state NEW -m multiport -p udp --dports 5404,5405 -j ACCEPT
```

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

```
iptables -I OUTPUT -m addrtype --dst-type MULTICAST -m state --state NEW -m multiport -p udp --dports 5404,5405 -j ACCEPT
```

For dlm (Distributed Lock Manager):

```
iptables -I INPUT -m state --state NEW -p tcp --dport 21064 -j ACCEPT
```

```
iptables -I OUTPUT -m state --state NEW -p tcp --dport 21064 -j ACCEPT
```

For ricci (remote agent):

```
iptables -I INPUT -m state --state NEW -p tcp --dport 11111 -j ACCEPT
```

```
iptables -I OUTPUT -m state --state NEW -p tcp --dport 11111 -j ACCEPT
```

For modclusterd (part of Conga remote agent):

```
iptables -I INPUT -m state --state NEW -p tcp --dport 16851 -j ACCEPT
```

```
iptables -I OUTPUT -m state --state NEW -p tcp --dport 16851 -j ACCEPT
```

For fenced (SMAPI TCP/IP-based fence agent):

```
iptables -I OUTPUT -m state --state NEW -p tcp --dport 44444 -j ACCEPT
```

For igmp (Internet Group Management Protocol):

```
iptables -I INPUT -p igmp -j ACCEPT
```

After executing these commands, run the following command to save the current configuration for the changes to be persistent during reboot.

```
service iptables save ; service iptables restart
```

4.13 Installing the SNA HAO luci code on CMANAGER

```
yum install luci
```

```
service luci start
```

4.14 Configure Administrator User on luci

Add a user to the system that you will use for cluster administration:

```
> useradd clusteradm
> passwd clusteradm
Changing password for user clusteradm.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
```

Enable luci to start at boot time:

```
chkconfig --levels 345 luci on
```

Start luci manually the first time:

```
service luci start
```

You should see something similar to this:

```
Adding following auto-detected host IDs (IP addresses/domain names),
corresponding to 'cmanager.devlab.sinenomine.net' address, to the
configuration of self-managed certificate
`/var/lib/luci/etc/cacert.config' (you can change them by editing
`/var/lib/luci/etc/cacert.config', removing the generated certificate
`/var/lib/luci/certs/host.pem' and restarting luci):
(none suitable found, you can still do it manually as mentioned
above)
```

```
Generating a 2048 bit RSA private key
writing new private key to '/var/lib/luci/certs/host.pem'
Starting saslauthd: [ OK ]
Start luci...
```

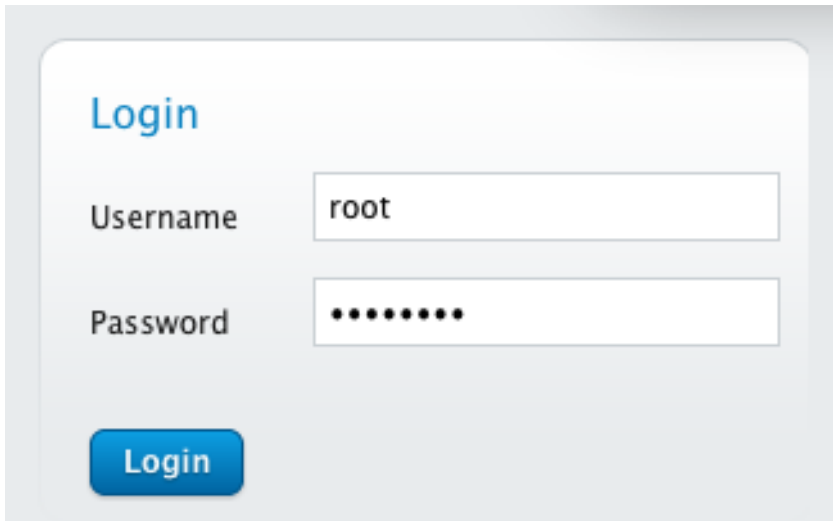
Point your web browser to <https://cmanger.devlab.sinenomine.net:8084>
(or equivalent) to access luci

Connect to luci by pointing your browser at <https://cmanger.devlab.sinenomine.net:8084>

Note: Problems have been experienced when using Internet Explorer 8 at the 8.0.7601.17514 level. It has been successfully tested with IE8 8.0.6001.18702, IE10, Firefox, Chrome and Safari.

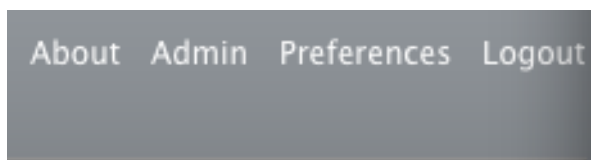
The first time you use luci, logon as root with the current root password. Your first task is to create another user that you will then use for all cluster administration.

Figure 2: luci login screen

A login form titled "Login" in blue text. It contains two input fields: "Username" with the text "root" and "Password" with ten black dots. Below the fields is a blue "Login" button.

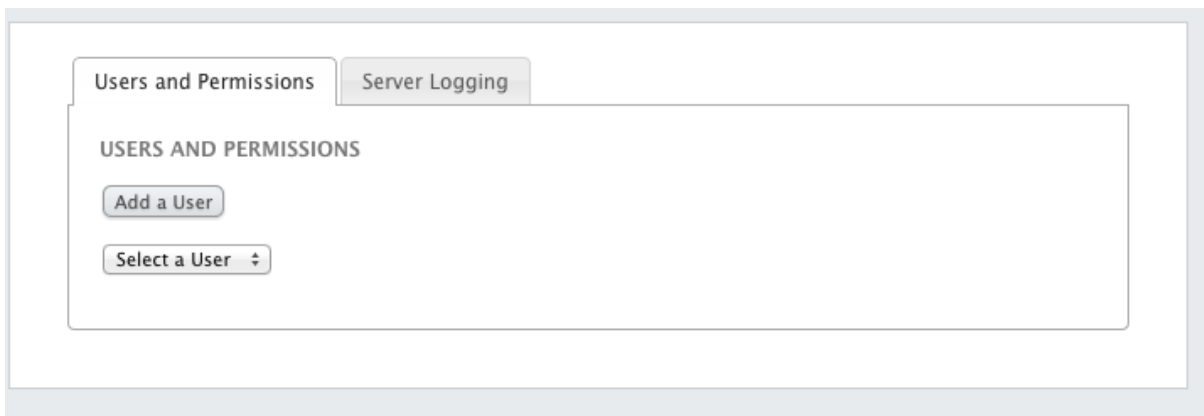
Once logged in, choose the “Admin” link at the top right of the window.

Figure 3: luci Action menu



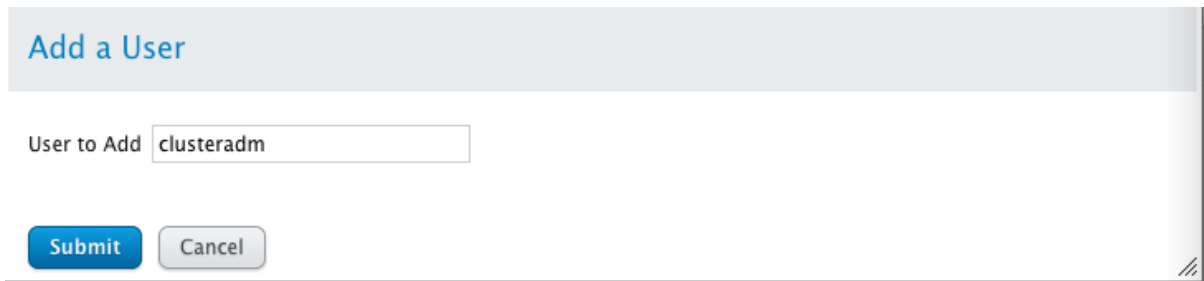
Click on “Add a User”

Figure 4: luci login screen

A web interface for "Users and Permissions". It has two tabs: "Users and Permissions" (active) and "Server Logging". Below the tabs is a section titled "USERS AND PERMISSIONS" containing an "Add a User" button and a "Select a User" dropdown menu.

Select a user name to add:

Figure 5: luci Add clusteradm user

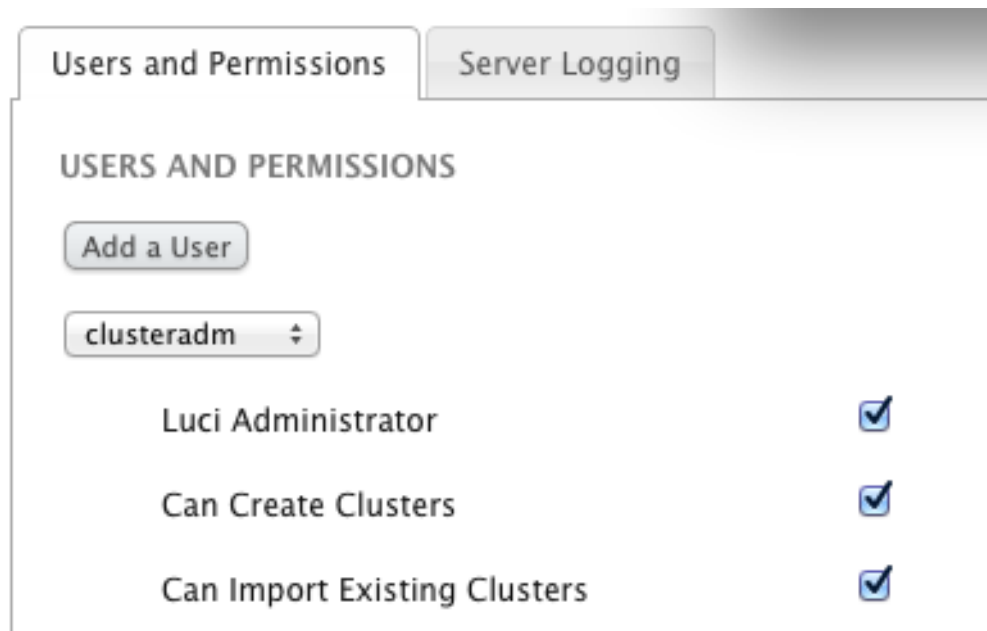


Add a User

User to Add

Give this user full administrator rights to luci and click the submit button:

Figure 6: luci Define clusteradm user privileges



Users and Permissions | **Server Logging**

USERS AND PERMISSIONS

Luci Administrator	<input checked="" type="checkbox"/>
Can Create Clusters	<input checked="" type="checkbox"/>
Can Import Existing Clusters	<input checked="" type="checkbox"/>

Choose logout from the top right and log back in as clusteradm using the password created with the passwd command above.

4.15 Installing the SNA HAO ricci code on CNODE1 and CNODE2

```
yum install -y ricci
```

```
chkconfig --levels 345 ricci start
service ricci start
Starting oddjobd: [ OK ]
generating SSL certificates... done
Generating NSS database... done
Starting ricci: [ OK ]
```

Set a password for ricci user. The luci tool will ask you for the password so that it can administer the nodes:

```
passwd ricci
```

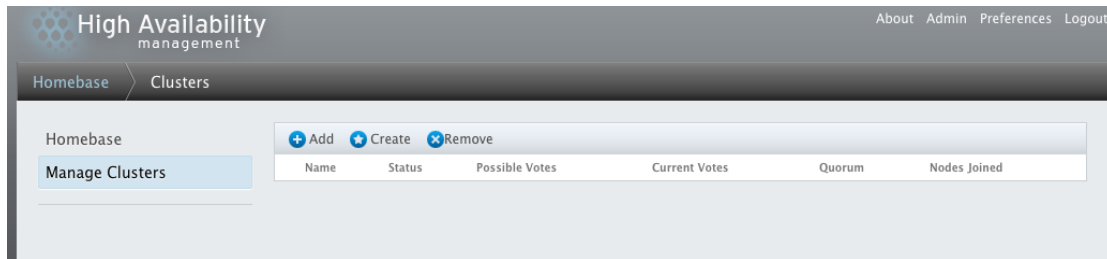
5 Creating the Cluster

5.1.1 Task: Create Cluster

Use luci to install the SNA HAO Code on Each Cluster Node

Click on the “Manage Clusters” item on the left of the screen and click the “Create” button:

Figure 7: luci Create cluster



5.2 Add CNODE1 and CNODE2 to Cluster

Define the name of the cluster and add nodes CNODE1 and CNODE2 to it. Check the “Download Packages” radio button, and check the “Reboot Nodes” and “Enable Shared” boxes before clicking on the “Create Cluster” button.

Figure 8: luci Add nodes to cluster

Create New Cluster

Cluster Name

☐ Use the Same Password for All Nodes

Node Name	Password	Ricci Hostname	Ricci Port
CNODE1	CNODE1	11111 ✕
CNODE2	CNODE2	11111 ✕

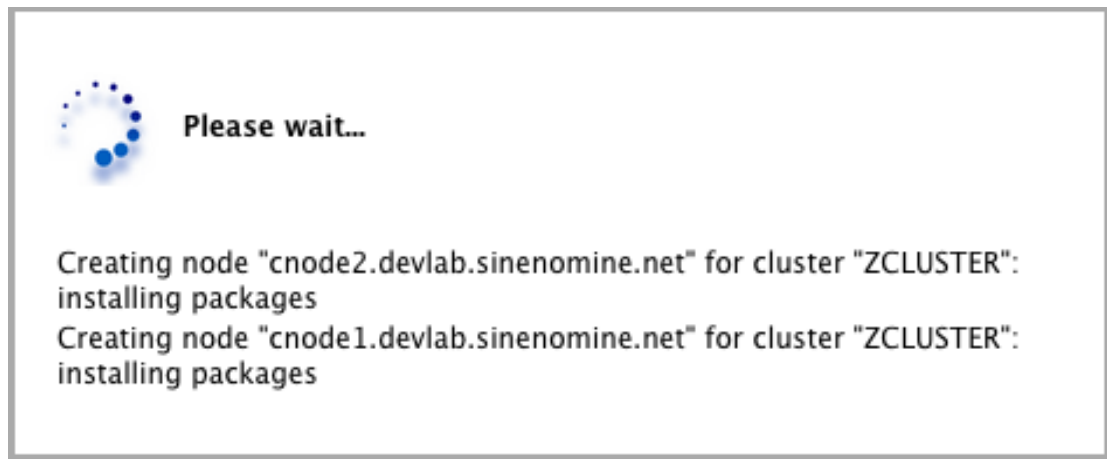
☒ Download Packages
☐ Use Locally Installed Packages

☒ Reboot Nodes Before Joining Cluster

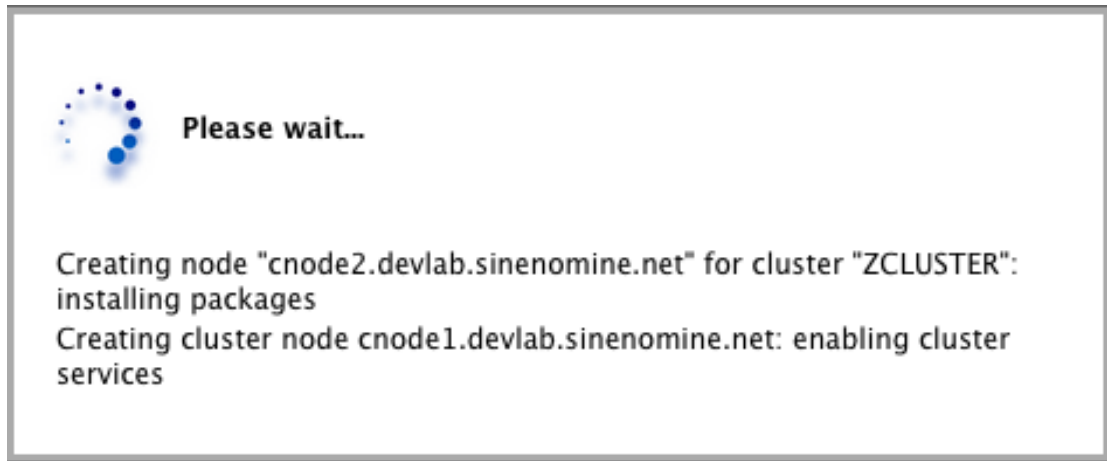
☒ Enable Shared Storage Support

During the cluster creation process you will see messages such as the following:

Figure 9: luci Cluster creation progress screen

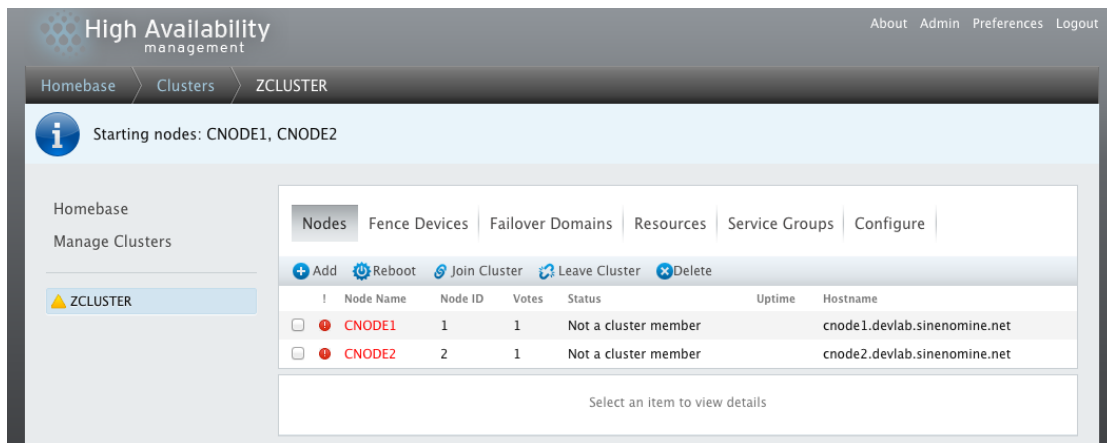


Eventually, cluster packages will be installed, a node enabled and rebooted:



When the cluster has been created the web page will report that the nodes have been added.

Figure 10: luci Cluster nodes added



This completes the installation of the HAO software on CNODE1 and CNODE2.

5.3 Task: Add a Fence Device

Fence device(s) are required so that failing nodes may be recovered. Click on the “Fence Devices” item and then click on the “Add” button.

Figure 11: luci Add fence device

Name	Fence Type	Nodes Using
No item to display		

Choose the “IBM z/VM – SNA over TCP/IP” menu item from the drop down list, give it a name, and provide the relevant information. Note the “SNA Authorized User Name” is the name defined in “4.4.3 Task: Edit VSMWORK1 AUTHLIST” on page 4-5 and its password is the one defined when the virtual machine was created.

Add Fence Device (Instance)

Fence Type: IBM z/VM – SNA over TCP/IP

Name: ZVMPOWER

SNA Virtual Machine Host IP Name/Address: vm.devlab.sinenomine.net

SNA Authorized User Name: CMANAGER

SNA Authorized User Password:

Submit Cancel

Figure 12: luci Add fence instance

5.4 Task: Add Fence Methods to CNODE1 and CNODE2

Figure 13: luci Add fence methods to nodes

+

Add

⚙

Reboot

🔗

Join Cluster

🔗

Leave Cluster

✖

Delete

!	Node Name	Node ID	Votes	Status	Uptime	Hostname
<input type="checkbox"/>	<div>CNODE1</div>	1	1	Not a cluster member		cnode1.devlab.sinenomine.net
<input type="checkbox"/>	<div>CNODE2</div>	2	1	Not a cluster member		cnode2.devlab.sinenomine.net

CNODE1

Status Not a cluster member

✖

🔗

🔗

⚙

Properties

Update Properties

Number of votes

1

ricci host

cnode1.devlab.sineno

ricci port

11111

Services

Failover Domains

Priority

Fence Devices

Method

Add Fence Method

Cluster Daemons

	Status
cman	Not running
rgmanager	Not running
ricci	Running
modclusterd	Running
clvmd	Not running

Choose a name for this instance of a fence method:

Figure 14: luci Choosing a name for fence method instance

Add Fence Method to Node

Method Name:

///

Add an instance of this method:

Figure 15: luci Adding an instance of fence method

Fence Devices

Method	
ZVMFENCE	Remove

Name Type/Values

Choose the fence device created earlier:

Figure 16: luci Choosing an existing fence device

Add Fence Device (Instance)

✓ -- Select a Fence Device --

ZVMPOWER (IBM z/VM - SMAPI over TCP/IP)

///

Next, provide parameters for this instance:

Figure 17: luci Providing parameters to fence instance

Add Fence Device (Instance)

ZVMPOWER (IBM z/VM - SMAPI over TCP/IP) ▾

Target Virtual Machine to Fence:

Shutdown timeout (optional):

Repeat this process for CNODE2, except specifying CNODE2 as the target virtual machine name.

When configuration is complete the “Nodes” display should look something like this:

Figure 18: luci Fence definition completed

	Node Name	Node ID	Votes	Status	Uptime	Hostname
<input type="checkbox"/>	CNODE1	1	1	Cluster Member	00:04:01:33	cnode1.devlab.sinenomine.net
<input type="checkbox"/>	CNODE2	2	1	Cluster Member	00:00:07:37	cnode2.devlab.sinenomine.net

5.5 Task: Additional Considerations for SSI Environments

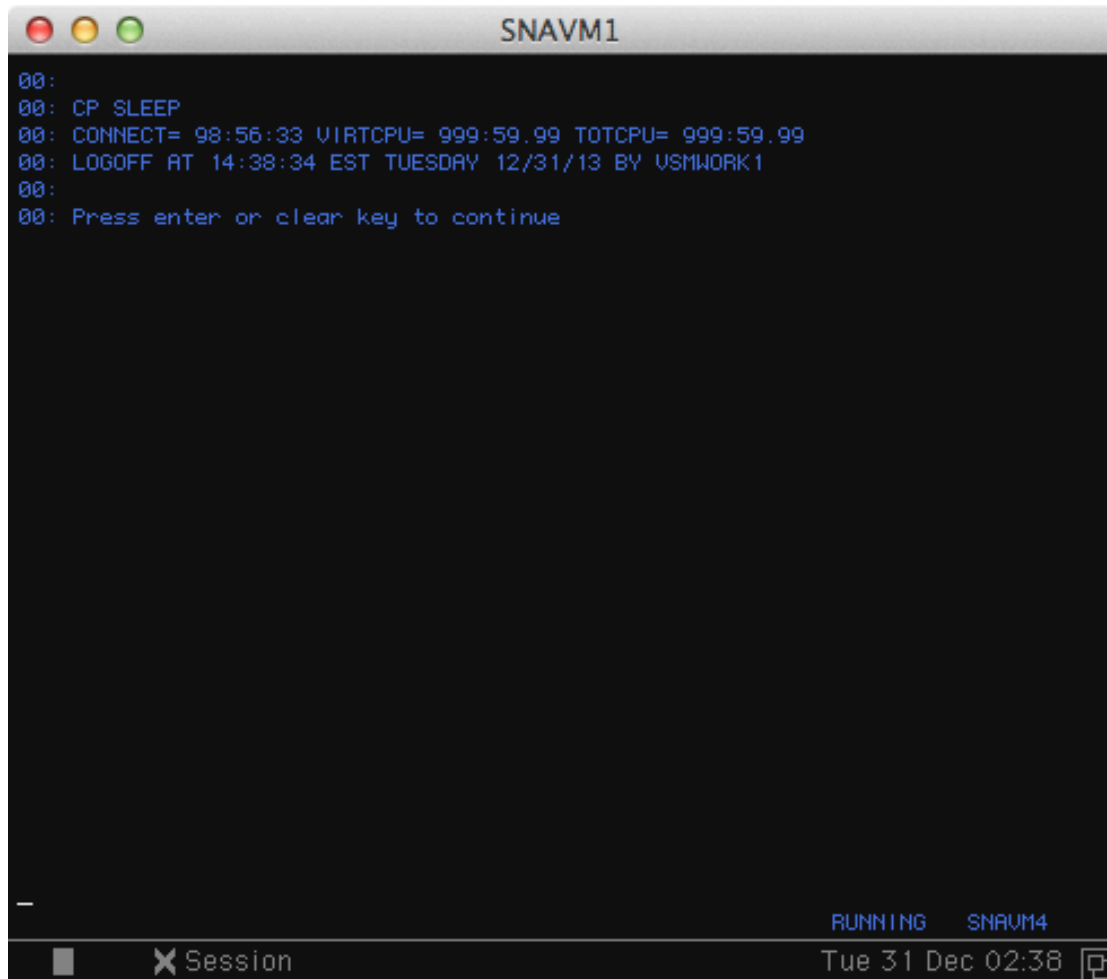
If running in a SSI environment, you must define a fence device for each VM system participating in the SSI cluster. For any HAO cluster running on these VM systems, you must include all the fence device definitions in the HAO cluster configuration for that cluster. HAO will try all the fence devices in order of definition until it gets a positive response that action was taken.

5.6 Task: Testing Fencing

With both nodes up and running, test the fencing operation by placing CNODE1 into a non-responsive state and have CNODE2 invoke the fence agent to cause CNODE1 to be forced off the system and logged back on.

Logon to the user CNODE1 via TN3270 and then enter #CP SLEEP on the console. This will stop any Linux activity and CNODE2 will soon detect CNODE1 being unresponsive.

Figure 19: Testing fencing



On CNODE2, the fence daemon log file (/var/log/cluster/fenced.log) will contain entries similar to these:

Figure 20: Testing fencing – node log contents

```

Dec 31 14:38:18 fenced cluster node 1 removed seq 524
Dec 31 14:38:23 fenced fenced:daemon conf 1 0 1 memb 2 join left 1
Dec 31 14:38:23 fenced fenced:daemon ring 2:528 1 memb 2
Dec 31 14:38:23 fenced fenced:default conf 1 0 1 memb 2 join left 1
Dec 31 14:38:23 fenced add_change cg 10 remove nodeid 1 reason 3
Dec 31 14:38:23 fenced add_change cg 10 m 1 j 0 r 1 f 1
Dec 31 14:38:23 fenced add_victims node 1
Dec 31 14:38:23 fenced check_ringid cluster 528 cpg 1:524
Dec 31 14:38:23 fenced fenced:default ring 2:528 1 memb 2
Dec 31 14:38:23 fenced check_ringid done cluster 528 cpg 2:528
Dec 31 14:38:23 fenced check_quorum done
Dec 31 14:38:23 fenced send_start 2:10 flags 2 started 9 m 1 j 0 r 1 f 1
Dec 31 14:38:23 fenced receive_start 2:10 len 192
Dec 31 14:38:23 fenced match_change 2:10 matches cg 10
Dec 31 14:38:23 fenced wait_messages cg 10 got all 1
Dec 31 14:38:23 fenced set_master from 2 to complete node 2
  
```

```

Dec 31 14:38:23 fenced delay post_fail_delay 0 quorate_from_last_update 0
Dec 31 14:38:23 fenced CNODE1 not a cluster member after 0 sec post_fail_delay
Dec 31 14:38:23 fenced fencing node CNODE1
Dec 31 14:38:38 fenced fence CNODE1 dev 0.0 agent fence_zvmip result: success
Dec 31 14:38:38 fenced fence CNODE1 success
Dec 31 14:38:38 fenced send_victim_done cg 10 flags 2 victim nodeid 1
Dec 31 14:38:38 fenced send_complete 2:10 flags 2 started 9 m 1 j 0 r 1 f 1
Dec 31 14:38:38 fenced receive_victim_done 2:10 flags 2 len 80
Dec 31 14:38:38 fenced receive_victim_done 2:10 remove victim 1 time
1388518718 how 1
Dec 31 14:38:38 fenced receive_complete 2:10 len 192

```

The z/VM operator log should show something similar to this:

Figure 21: Testing fencing – z/VM log contents

```

14:38:23 CNODE2 *8 dlm: closing connection to node 1
14:38:24 CNODE2 *8 GFS2: fsid=ZCLUSTER:vol1.1: jid=0: Trying to acquire journal lock...
14:38:36 OPERATOR *3 GRAF L0003 LOGOFF AS CNODE1 USERS = 59 FORCED BY VSMWORK1
14:38:36 VSMWORK1 *8 14:38:36 GRAF L0003 LOGOFF AS CNODE1 USERS = 59 FORCED BY
VSMWORK1
14:38:37 OPERATOR *3 AUTO LOGON *** CNODE1 USERS = 60 BY VSMWORK1

```

CNODE1 will now reboot and automatically rejoin the cluster.

6 Configuring Cluster Resources and Services

6.1 Formatting Shared GFS2 Disk

6.1.1 Task: Activate Shared GFS2 Disk on CNODE1 and CNODE2

If the shared gfs2 disk was not added at RHEL installation time, then make them available now on both nodes:

```
cio_ignore -R
chccwdev -e 0.0.0153
```

Update /etc/zipl.conf to make these devices available at boot time:

```
- parameters="root=/dev/mapper/vg_system-root rd_NO_LUKS
rd_LVM_LV=vg_system/root LANG=en_US.UTF-8 rd_DASD=0.0.0150
rd_DASD=0.0.0151 rd_DASD=0.0.0152 KEYTABLE=us cio_ignore=all,!0.0.0009
rd_NO_MD crashkernel=auto SYSFONT=latarcyrheb-sun16 rd_NO_DM"

+ parameters="root=/dev/mapper/vg_system-root rd_NO_LUKS
rd_LVM_LV=vg_system/root LANG=en_US.UTF-8 rd_DASD=0.0.0150
rd_DASD=0.0.0151 rd_DASD=0.0.0152 rd_DASD=0.0.0153 KEYTABLE=us
cio_ignore=all,!0.0.0009 rd_NO_MD crashkernel=auto SYSFONT=latarcyrheb-
sun16 rd_NO_DM"
```

Run zipl to make the changes permanent.

6.1.2 Task: Format and Partition Shared Cluster Disks on CNODE1

On CNODE1 Format and partition the two shared disks (gfs2 and the optional quota disk):

```
dasdfmt -b 4096 -y /dev/dasdd
fdasd -a /dev/dasdd
```

6.2 Setting Up a GFS2 Filesystem

6.2.1 Task: Create Physical Volume

```
pvccreate /dev/dasdd1
```

6.2.2 Task: Create Cluster Volume Group

```
vgcreate -cy vg_cluster /dev/dasdd1
```

6.2.3 Task: Make Volume Group Visible from CNODE2

As root on CNODE2:

```
vgchange -ay vg_cluster
```

6.2.4 Task: Create Logical Volume

```
lvcreate -L 500m -n ha_lv vg_cluster
```

6.2.5 Task: Create GFS2 Filesystem

```
mkfs -t gfs2 -j 2 -t ZCLUSTER:vol1 /dev/mapper/vg_cluster-ha_lv
```

For a small test file system you may want to reduce the overhead used by journals and resource groups. The above example will result in the use of around 259M, the example below only 25M:

```
mkfs -t gfs2 -j 2 -J 16 -r 32 -t ZCLUSTER:vol1 /dev/mapper/vg_cluster-ha_lv
```

6.3 Creating HA Web Server on CNODE1 and CNODE2

6.3.1 Task: Install Apache

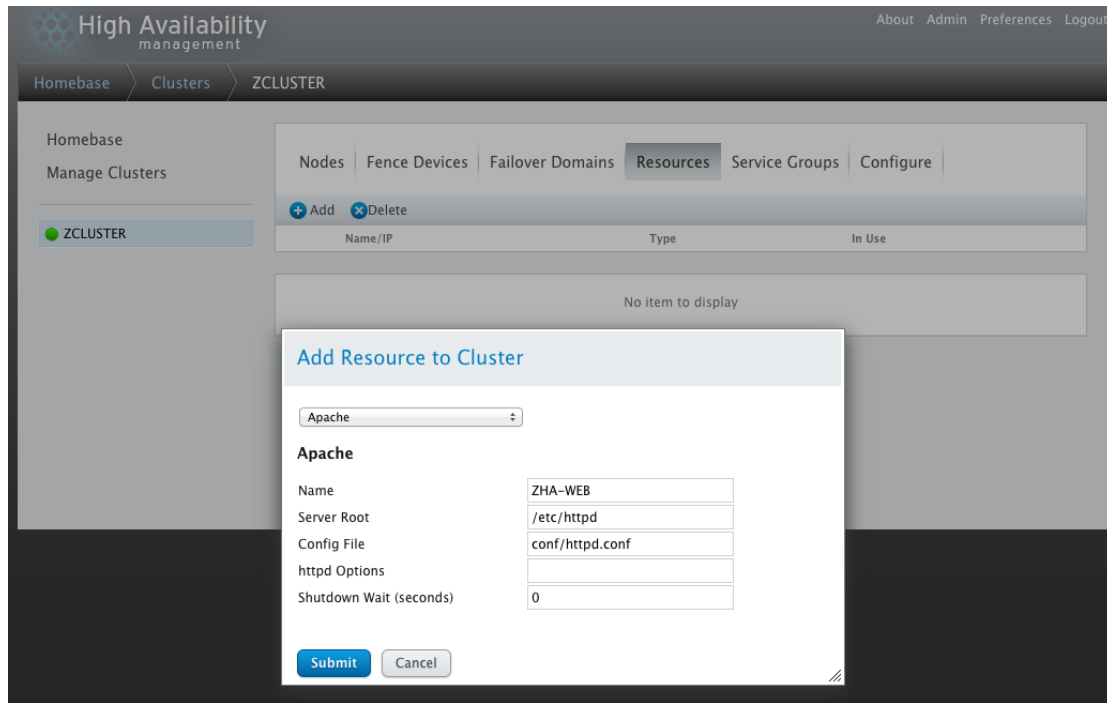
```
yum install -y httpd
```

6.4 Create Cluster Resources

6.4.1 Task: Create Web Server Resource

Click on the “Add Resource” button and select “Apache” from the drop down list. Specify any required parameters (the defaults are fine). When done, click “Submit”.

Figure 22: luci Adding Apache Resource



6.4.2 Task: Create GFS2 Resource

Click on the “Add Resource” and select “GFS/GFS2” from the drop-down menu. Specify the required parameters and click “Submit”.

Figure 23: luci Adding GFS2 Resource

Add Resource to Cluster

GFS/GFS2

GFS2

Name

ZHA-GFS

Mount Point

/var/www/html

Device, FS Label, or UUID

/dev/mapper/vg_cluster-ha_lv

Filesystem Type

GFS2

Mount Options

Filesystem ID (optional)

Force Unmount

☐

Enable NFS daemon and lockd workaround

☐

Reboot Host Node if Unmount Fails

☐

Submit

Cancel

6.4.3 Task: Create IP Address Resource

Add an IP address that you have reserved for the Web Service. This is a “virtual” address as it will be associated with the service and not a physical host. Note the netmask needs to be specified as the number of bits used for the host portion of the address (for example, 24) not as an “AND” mask like 255.255.255.0:

Figure 24: luci Adding IP Address Resource

Add Resource to Cluster

IP Address

IP Address

IP Address: 172.17.16.3

Netmask Bits (optional): 24

Monitor Link: ☒

Disable Updates to Static Routes: ☐

Number of Seconds to Sleep After Removing an IP Address: 10

Submit Cancel

6.5 Task: Create a Cluster Failover Domain

Create a failover domain consisting of the two nodes.

Figure 25: luci Creating a failover domain

High Availability management

About Admin Preferences Logout

Homebase > Clusters > ZCLUSTER

Homebase
Manage Clusters

ZCLUSTER

Nodes Fence Devices **Failover Domains** Resources Service Groups Configure

+ Add - Delete

Add Failover Domain to Cluster

Name: ZHA_FAILOVER

☐ Prioritized Order the nodes to which services failover.

☐ Restricted Service can run only on nodes specified.

☒ No Failback Do not send service back to 1st priority node when it becomes available again.

Member	Priority
CNODE1	
CNODE2	

Create Cancel

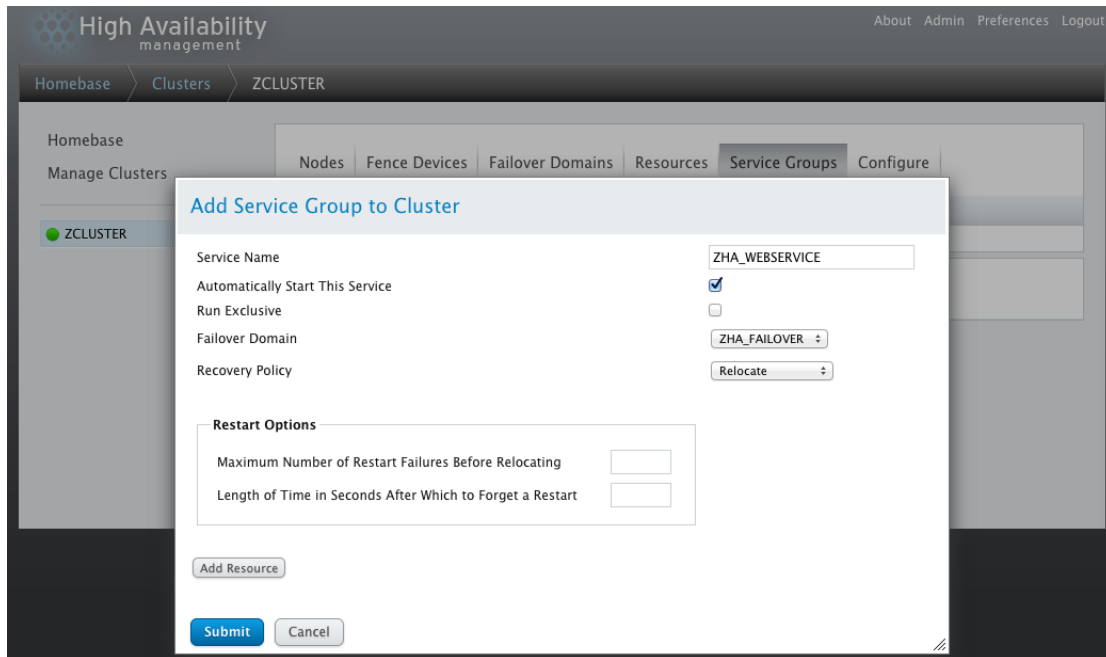
6.6 Create a Cluster Service Group

Create a service that will be run on either node and will failover to the other node in the case of failure. The service will consist of the resources created in the previous steps.

6.6.1 Task: Create the Service Group

Create a service group to represent a group of resources used to implement a service:

Figure 26: luci Creating a service group

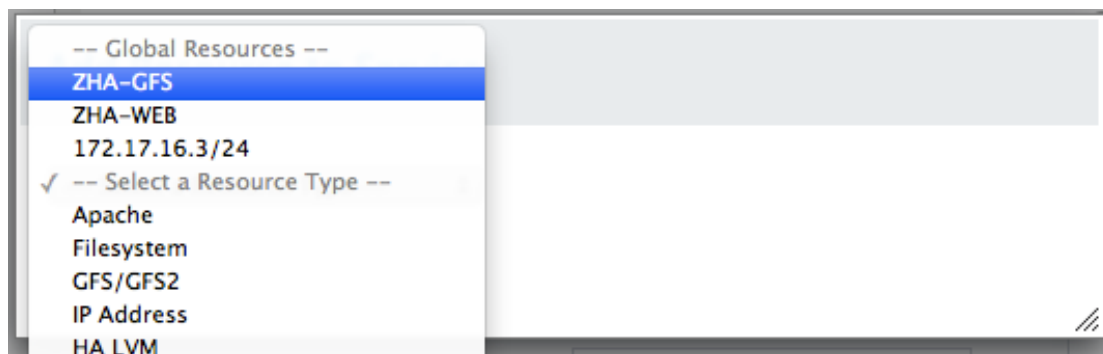


Click on the “Add Resource” button to add the following resources.

6.6.2 Task: Add GFS2 Resource

Add the GFS resource created earlier to this service group.

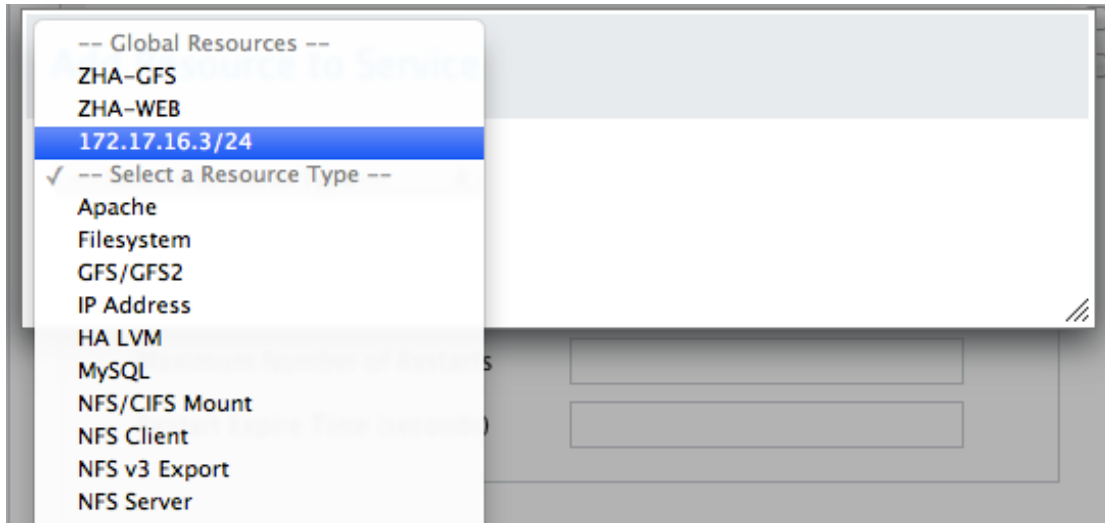
Figure 27: luci Adding GFS2 resource to the service group



6.6.3 Task: Add IP Address Resource

Add the IP address resource created earlier to this service group:

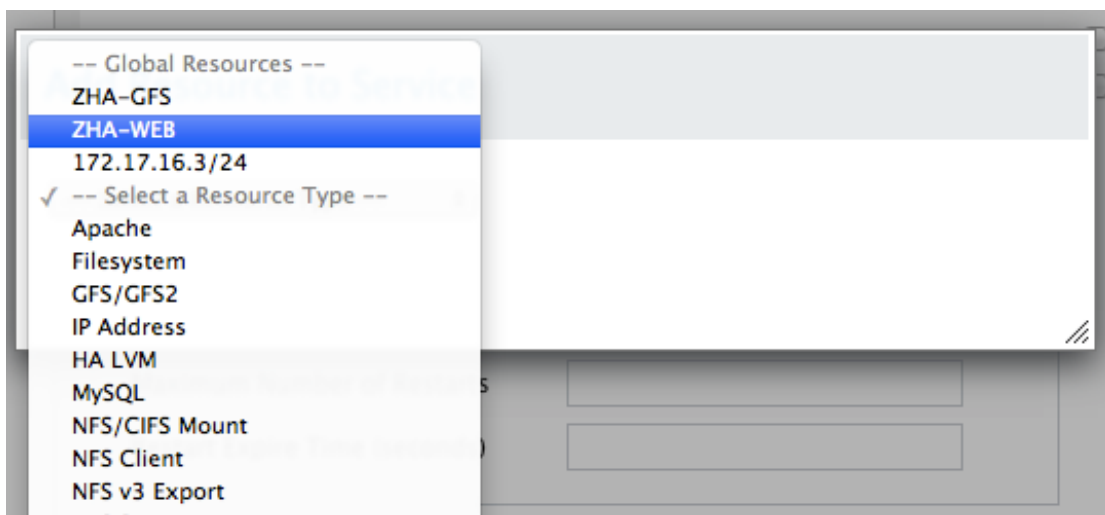
Figure 28: luci Adding IP resource to the service group



6.6.4 Task: Add Web Service Resource

Add the Web Server resource created earlier to the service group:

Figure 29: luci Adding Web Service resource to the service group

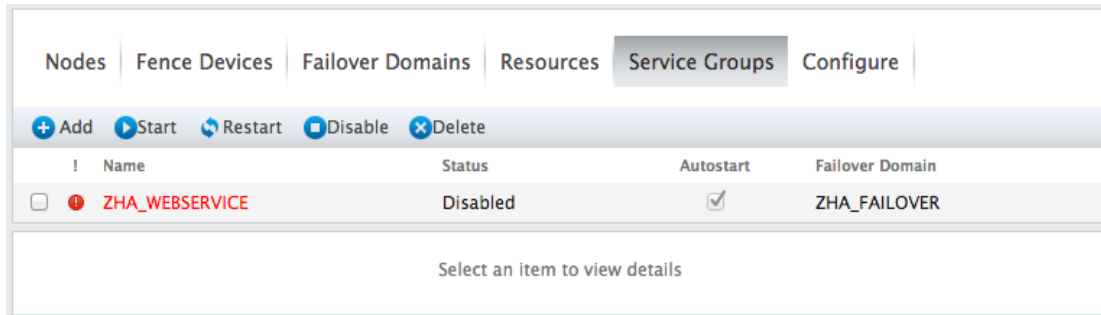


When these resources have been added click the “Submit” button to complete the creation of the service group.

6.7 Task: Starting the Service

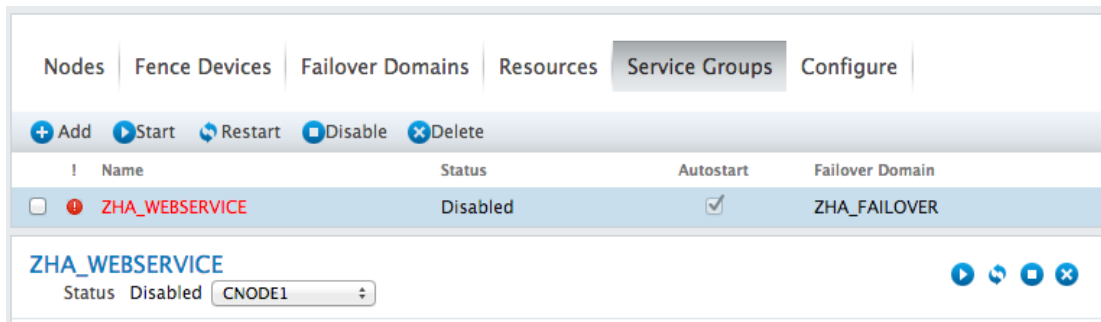
To start the service you can check the box and click the “Start” link:

Figure 30: luci Starting the service using Start link



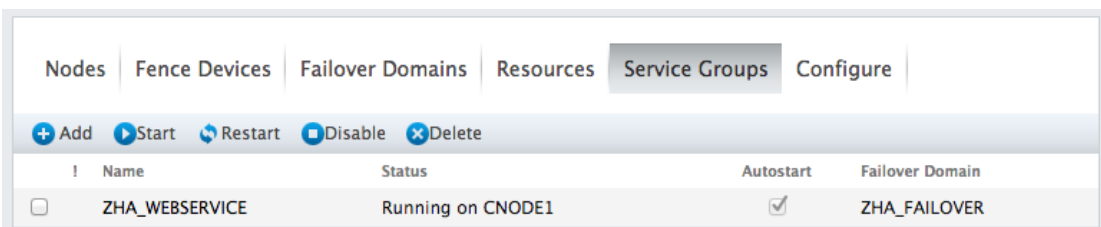
Alternatively, click on the service name to start the service on a specific node.

Figure 31: luci Starting the service on a specific node



After the service starts you should see something like this:

Figure 32: luci Result of starting the service



6.8 Task: Create Home Page

Create a simple test page: /var/www/html/index.html

```
<html>
<title>Cluster Test Page</title>
<body>
<h1>Cluster Test Page</h1>
</body>
</html>
```

6.9 Task: Point Browser at HA IP Address

Point your browser at the IP address defined in the resources (172.17.16.3):

Figure 33: Accessing service via Browser



Figure 1: Example of XML-based Cluster Definition File (created by luci)

```

<?xml version="1.0"?>
<cluster config_version="29" name="ZCLUSTER">
  <clusternodes>
    <clusternode name="CNODE1" nodeid="1">
      <fence>
        <method name="ZVMFENCE">
          <device name="ZVMPPOWER" port="CNODE1"/>
        </method>
      </fence>
    </clusternode>
    <clusternode name="CNODE2" nodeid="2">
      <fence>
        <method name="ZVMFENCE">
          <device name="ZVMPPOWER" port="CNODE2"/>
        </method>
      </fence>
    </clusternode>
  </clusternodes>
  <cman expected_votes="3"/>
  <fencedevices>
    <fencedevice agent="fence_zvmip" ipaddr="vm.devlab.sinenomine.net" login="CMANAGER" name="ZVMPPOWER" passwd="XXXXXXX"/>
  </fencedevices>
</rm>
  <failoverdomains>
    <failoverdomain name="ZHA_FAILOVER" nofailback="1">
      <failoverdomainnode name="CNODE1"/>
      <failoverdomainnode name="CNODE2"/>
    </failoverdomain>
  </failoverdomains>
  <resources>
    <clusterfs device="/dev/mapper/vg_cluster-ha_lv" fsid="37908" fstype="gfs2" mountpoint="/var/www/html" name="ZHA-GFS"/>
    <apache config_file="conf/httpd.conf" name="ZHA-WEB" server_root="/etc/httpd" shutdown_wait="0"/>
    <ip address="172.17.16.3/24" sleep_time="10"/>
  </resources>
  <service domain="ZHA_FAILOVER" name="ZHA_WEBSERVICE" recovery="relocate">
    <clusterfs ref="ZHA-GFS"/>
    <ip ref="172.17.16.3/24"/>
    <apache ref="ZHA-WEB"/>

```

```
<logging logfile_priority="warning" syslog_priority="warning">  
  <logging_daemon logfile_priority="warning" name="corosync" syslog_priority="warning"/>  
</logging>  
<quorumd label="QDISK"/>  
</cluster>
```

6.10 Task: Failover Web Service to CNODE2

Kill the Web Service on CNODE1

```
killall httpd
```

You should see the following appear in the rgmanager.log file:

```
Dec 27 12:17:07 rgmanager [apache] Verifying Configuration Of apache:ZHA-WEB
Dec 27 12:17:10 rgmanager [apache] Checking Syntax Of The File /etc/httpd/conf/httpd.conf
Dec 27 12:17:13 rgmanager [apache] Checking Syntax Of The File /etc/httpd/conf/httpd.conf > Succeed
Dec 27 12:17:16 rgmanager [apache] Monitoring Service apache:ZHA-WEB
Dec 27 12:17:19 rgmanager [apache] Checking Existence Of File /var/run/cluster/apache/apache:ZHA-WEB.pid
[apache:ZHA-WEB] > Failed
Dec 27 12:17:22 rgmanager [apache] Monitoring Service apache:ZHA-WEB > Service Is Not Running
Dec 27 12:17:22 rgmanager status on apache "ZHA-WEB" returned 7 (unspecified)
Dec 27 12:17:22 rgmanager Stopping service service:ZHA_WEBSERVICE
Dec 27 12:17:27 rgmanager [apache] Verifying Configuration Of apache:ZHA-WEB
Dec 27 12:17:30 rgmanager [apache] Checking Syntax Of The File /etc/httpd/conf/httpd.conf
Dec 27 12:17:34 rgmanager [apache] Checking Syntax Of The File /etc/httpd/conf/httpd.conf > Succeed
Dec 27 12:17:38 rgmanager [apache] Stopping Service apache:ZHA-WEB
Dec 27 12:17:41 rgmanager [apache] Checking Existence Of File /var/run/cluster/apache/apache:ZHA-WEB.pid
[apache:ZHA-WEB] > Failed - File Doesn't Exist
Dec 27 12:17:44 rgmanager [apache] Stopping Service apache:ZHA-WEB > Succeed
Dec 27 12:17:48 rgmanager [ip] Removing IPv4 address 172.17.16.3/24 from eth0
Dec 27 12:18:04 rgmanager [clusterfs] Not unmounting /dev/dm-6 (clustered file system)
Dec 27 12:18:04 rgmanager Service service:ZHA_WEBSERVICE is recovering
```

On CNODE2, the rgmanager.log file should contain:

```

Dec 27 12:18:05 rgmanager Recovering failed service service:ZHA_WEBSERVICE
Dec 27 12:18:12 rgmanager [clusterfs] mounting /dev/dm-6 on /var/www/html
Dec 27 12:18:16 rgmanager [clusterfs] mount -t gfs2 /dev/dm-6 /var/www/html
Dec 27 12:18:26 rgmanager [ip] Link for eth0: Detected
Dec 27 12:18:29 rgmanager [ip] Adding IPv4 address 172.17.16.3/24 to eth0
Dec 27 12:18:32 rgmanager [ip] Ping addr 172.17.16.3 from dev eth0
Dec 27 12:18:38 rgmanager [ip] Sending gratuitous ARP: 172.17.16.3 02:00:00:00:00:6d brd ff:ff:ff:ff:ff:ff
Dec 27 12:18:46 rgmanager [apache] Verifying Configuration Of apache:ZHA-WEB
Dec 27 12:18:49 rgmanager [apache] Checking Syntax Of The File /etc/httpd/conf/httpd.conf
Dec 27 12:18:53 rgmanager [apache] Checking Syntax Of The File /etc/httpd/conf/httpd.conf > Succeed
Dec 27 12:18:56 rgmanager [apache] Monitoring Service apache:ZHA-WEB
Dec 27 12:18:59 rgmanager [apache] Checking Existence Of File /var/run/cluster/apache/apache:ZHA-WEB.pid
[apache:ZHA-WEB] > Failed
Dec 27 12:19:02 rgmanager [apache] Monitoring Service apache:ZHA-WEB > Service Is Not Running
Dec 27 12:19:05 rgmanager [apache] Starting Service apache:ZHA-WEB
Dec 27 12:19:08 rgmanager [apache] Looking For IP Addresses
Dec 27 12:19:14 rgmanager [apache] 1 IP addresses found for ZHA_WEBSERVICE/ZHA-WEB
Dec 27 12:19:19 rgmanager [apache] Looking For IP Addresses > Succeed - IP Addresses Found
Dec 27 12:19:23 rgmanager [apache] Checking: SHA1 checksum of config file /etc/cluster/apache/apache:ZHA-
WEB/httpd.conf
Dec 27 12:19:26 rgmanager [apache] Checking: SHA1 checksum > succeed
Dec 27 12:19:29 rgmanager [apache] Generating New Config File /etc/cluster/apache/apache:ZHA-WEB/httpd.conf
From /etc/httpd/conf/httpd.conf
Dec 27 12:19:35 rgmanager [apache] Generating New Config File /etc/cluster/apache/apache:ZHA-WEB/httpd.conf
From /etc/httpd/conf/httpd.conf > Succeed
Dec 27 12:19:40 rgmanager [apache] Starting Service apache:ZHA-WEB > Succeed
Dec 27 12:19:40 rgmanager Service service:ZHA_WEBSERVICE started
Dec 27 12:19:50 rgmanager [clusterfs] Checking fs "ZHA-GFS", level 0
Dec 27 12:19:54 rgmanager [ip] Checking 172.17.16.3/24, level 10
Dec 27 12:19:57 rgmanager [ip] 172.17.16.3/24 present on eth0
Dec 27 12:20:01 rgmanager [ip] Link for eth0: Detected
Dec 27 12:20:04 rgmanager [ip] Link detected on eth0
Dec 27 12:20:08 rgmanager [ip] Local ping to 172.17.16.3 succeeded

```



```
Dec 27 12:20:20 rgmanager [ip] Checking 172.17.16.3/24, Level 0
Dec 27 12:20:23 rgmanager [ip] 172.17.16.3/24 present on eth0
Dec 27 12:20:26 rgmanager [ip] Link for eth0: Detected
Dec 27 12:20:29 rgmanager [ip] Link detected on eth0
Dec 27 12:20:50 rgmanager [apache] Verifying Configuration Of apache:ZHA-WEB
Dec 27 12:20:53 rgmanager [apache] Checking Syntax Of The File /etc/httpd/conf/httpd.conf
Dec 27 12:20:57 rgmanager [apache] Checking Syntax Of The File /etc/httpd/conf/httpd.conf > Succeed
Dec 27 12:21:00 rgmanager [apache] Monitoring Service apache:ZHA-WEB
Dec 27 12:21:03 rgmanager [apache] Monitoring Service apache:ZHA-WEB > Service Is Running
```

6.11 Task: Point Browser at HA IP Address

Figure 34: Testing Failover

Test the1 failover by pointing the browser to the 172.17.16.3 address:



Cluster Test Page

7 Common Variations to This Design

7.1 Adding a Quorum Disk

Generally, for a two-node configuration it is not necessary to implement a quorum disk. However, it can be done. A quorum disk provides a mechanism for determining node-fitness in a cluster environment.

If you did not create the quorum disk in the earlier steps, then the following tasks will create the device and make it available.

7.1.1 Task: Add Volumes to User COMMON

If the quorum disk (COMMON 0200) was not defined when the userid was created...

7.1.2 Task: Activate Volume on CNODE1

Make the vmcp kernel module available:

```
modprobe vmcp
vmcp link \* 200 200 mw
```

Clear the I/O blacklist so that the device can be made visible to the operating system:

```
cio_ignore -R
```

Vary the device online:

```
chccwdev -e 0.0.0200
```

Edit /etc/zipl.conf to make the device available after each boot:

```
- parameters="root=/dev/mapper/vg_system-root rd_NO_LUKS
rd_LVM_LV=vg_system/root LANG=en_US.UTF-8 rd_DASD=0.0.0150
rd_DASD=0.0.0151 rd_DASD=0.0.0152 rd_DASD=0.0.0153 KEYTABLE=us
cio_ignore=all,!0.0.0009 rd_NO_MD crashkernel=auto SYSFONT=latarcyrhel
sun16 rd_NO_DM"
```

```
+ parameters="root=/dev/mapper/vg_system-root rd_NO_LUKS
rd_LVM_LV=vg_system/root LANG=en_US.UTF-8 rd_DASD=0.0.0150
rd_DASD=0.0.0151 rd_DASD=0.0.0152 rd_DASD=0.0.0153 rd_DASD=0.0.0200
KEYTABLE=us cio_ignore=all,!0.0.0009 rd_NO_MD crashkernel=auto
SYSFONT=latarcyrheb-sun16 rd_NO_DM"
```

Run zipl to re-write the bootstrap.

```
zipl
```

7.1.3 Task: Format and Partition New Volume on NODE1

```
dasdfmt -b 4096 -y /dev/dasde
fdasd -a /dev/dasde
```

7.1.4 Task: Make the Quorum Disk File System

7.1.5 Task: Creating Quorum Disk [Optional]

Create the quorum disk with the `mkqdisk` command. Take note of the `-l` option as it will be used to configure the disk in the cluster.

```
mkqdisk -c /dev/dasde1 -l QDISK
mkqdisk v3.0.12.1
```

```
Writing new quorum disk label 'QDISK' to /dev/dasde1.
WARNING: About to destroy all data on /dev/dasde1; proceed [N/y] ? y
Initializing status block for node 1...
Initializing status block for node 2...
Initializing status block for node 3...
Initializing status block for node 4...
Initializing status block for node 5...
Initializing status block for node 6...
Initializing status block for node 7...
Initializing status block for node 8...
Initializing status block for node 9...
Initializing status block for node 10...
Initializing status block for node 11...
Initializing status block for node 12...
Initializing status block for node 13...
Initializing status block for node 14...
Initializing status block for node 15...
Initializing status block for node 16...
```

7.1.6 Task: Activate Volume on CNODE2

Make the `vmcp` kernel module available (note the backslash is required to escape the `'*'` wildcard character):

```
modprobe vmcp
vmcp link \* 200 200 mw
```

Clear the I/O blacklist so that the device can be made visible to the operating system:

```
cio_ignore -R
```

Vary the device online:

```
chccwdev -e 0.0.0200
```

Edit /etc/zipl.conf to make the device available after each boot:

```
- parameters="root=/dev/mapper/vg_system-root rd_NO_LUKS
rd_LVM_LV=vg_system/root LANG=en_US.UTF-8 rd_DASD=0.0.0150
rd_DASD=0.0.0151 rd_DASD=0.0.0152 rd_DASD=0.0.0153 KEYTABLE=us
cio_ignore=all,!0.0.0009 rd_NO_MD crashkernel=auto SYSFONT=latarcyrheb-
sun16 rd_NO_DM"
```

```
+ parameters="root=/dev/mapper/vg_system-root rd_NO_LUKS
rd_LVM_LV=vg_system/root LANG=en_US.UTF-8 rd_DASD=0.0.0150
rd_DASD=0.0.0151 rd_DASD=0.0.0152 rd_DASD=0.0.0153 rd_DASD=0.0.0200
KEYTABLE=us cio_ignore=all,!0.0.0009 rd_NO_MD crashkernel=auto
SYSFONT=latarcyrheb-sun16 rd_NO_DM"
```

Run zipl to re-write the bootstrap.

```
zipl
```

7.1.7 Task: Define Quorum Disk on CMANAGER

Select the “Configure” tab and then select the “QDisk” option. Use the label data used on the mkqdisk command to specify the disk to the cluster configuration.

Figure 35: luci Defining a Quorum Disk

Nodes Fence Devices Failover Domains Resources Service Groups **Configure**

General Fence Daemon Network Redundant Ring **QDisk** Logging

Quorum Disk Configuration

☐ Do Not Use a Quorum Disk
☒ Use a Quorum Disk

Specify Physical Device

☒ By Device Label

☐ By Filesystem Path to Device (deprecated)

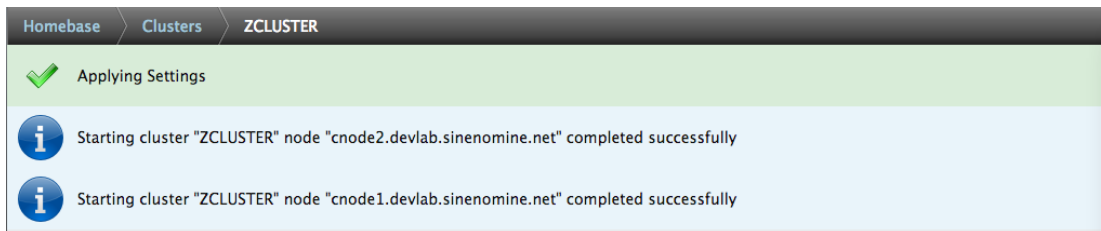
Heuristics

Path to Program	Interval	Score	TKO
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>

Minimum Total Score

After you click the “Apply” button, you should see the following:

Figure 36: luci Quorum Disk Created



7.2 Where to Find Out More Information

8 z/VM Fence Devices

There are two fence devices for use with RHEL HA on System z:

8.1 fence_zvm

`fence_zvm` is a Power Fencing agent used on a HA virtual machine in a System z z/VM cluster. It uses the SMAPI interface to deactivate an active image.

`fence_zvm` accepts options on the command line as well as from stdin. `fence_node` sends the options through stdin when it executes the agent. `fence_zvm` can be run by itself with command line options, which is useful for testing.

Table 6: fence_zvm parameters

Option	Description
Command line Options	
-o --action action	Fencing action: "off" - fence off device; "metadata" - display device metadata
-n --port target	Name of virtual machine to deactivate.
-h --help	Print out a help message describing available options, then exit.
-a --ip Server	Name of SMAPI server virtual machine. To be consistent with other fence agents this name is a little misleading: it is the name of the virtual machine not its IP address or hostname.
-T --timeout time	Timeout – currently ignored
--zvm sys	Name of z/VM system upon which the SMAPI server is running (optional – defaults to system on which node is running). This parameter is for future use only. Current AF_IUCV support in the kernel does not include distributed IUCV.
Standard Input Options	
agent = agent	This option is used by <code>fence_node(8)</code> and is ignored by <code>fence_zvm</code> .
action = action	Fencing action: "off" - fence off device; "metadata" - display device metadata
plug = target	Name of virtual machine to deactivate.
ipaddr = srv	Name of SMAPI server virtual machine. To be consistent with other

	fence agents this name is a little misleading: it is the name of the virtual machine not its IP address or hostname.
timeout = time	Timeout – currently ignored
zvm sys = system	Name of z/VM system upon which the SMAPI server is running (optional – defaults to system on which node is running) This parameter is for future use only. Current AF_IUCV support in the kernel does not include distributed IUCV.

This fence device uses the IUCV interface to SMAPI, which means it is not dependent on the network being available to perform its tasks. However, this also means that if network connectivity is lost to a virtual machine running this agent it will detect the loss of connection and assume that the other node has lost contact. Therefore, it will attempt to recycle the other node. At the same time other nodes that notice the loss of connectivity with this node will also perform the fencing action on the node. The result is that instead of one node being fenced both nodes get recycled.

To use this agent the z/VM SMAPI service needs to be configured to allow the virtual machine running this agent to connect to it and issue the `image_recycle` operation. This involves updating the VSMWORK1 AUTHLIST VMSYS:VSMWORK1. file. The entry should look something similar to this:

Column 1	Column 66	Column 131
V	V	V
XXXXXXXX	ALL	IMAGE_OPERATIONS

Where XXXXXXXX is the name of the virtual machine where the agent resides.

In addition, the VM directory entry that defines this virtual machine requires the IUCV ANY statement (or IUCV <userid of SMAPI Server>). This authorizes use of IUCV to connect to the SMAPI server.

8.2 fence_zvmip

fence_zvmip is an power Fencing agent used on a GFS virtual machine in a System z z/VM cluster. It uses the TCP/IP SMAPI interface to deactivate an active image. fence_zvmip accepts options on the command line as well as from stdin. fence_node sends the options through stdin when it execs the agent. fence_zvmip can be run by itself with command line options, which is useful for testing.

Table 7: fence_zvmip parameters

Option	Description
Command line Options	
-o --action action	Fencing action: "off" - fence off device; "metadata" - display device

	metadata
-n --port target	Name of virtual machine to deactivate.
-a --ip Server	IP Name or address z/VM system where SMAPI server resides.
-h --help	Print out a help message describing available options, then exit.
-u --username user	Name of an authorized SMAPI user
-p --password pass	Password of the authorized SMAPI user
-T --timeout time	Timeout – currently ignored
Standard Input Options	
agent = agent	This option is used by fence_node(8) and is ignored by fence_zvmip.
action = action	Fencing action: "off" - fence off device; "metadata" - display device metadata
plug = target	Name of virtual machine to deactivate.
ipaddr = Server	IP Name or address z/VM system where SMAPI server resides.
username = user	Name of SMAPI server virtual machine.
password = pass	Password of the authorized SMAPI user
timeout = time	Timeout – currently ignored

To use this agent the z/VM SMAPI service needs to be configured to allow the virtual machine running this agent to connect to it and issue the `image_recycle` operation. This involves updating the VSMWORK1 AUTHLIST VMSYS:VSMWORK1. file. The entry should look something similar to this:

Column 1	Column 66	Column 131
V	V	V
XXXXXXXX	ALL	IMAGE_OPERATIONS

Where XXXXXXXX is the name of the userid used as the “authuser” parameter for the agent.

When using the IP fence device, we recommend that a special non-privileged user be set up that can be used by the fence device(s). This avoids having to use the userid and password of the node itself in the configuration file. For example, use your directory manager (or hand-edit the directory) to create a user called HAO which has no disks, class G privileges, minimal memory (e.g. 8K) and does not IPL an operating system. Use this id and its password when configuring the fence devices for a given z/VM system.

9 Other Files

9.1 Virtual Machine A Disks

The following files will reside on the “A” disk of the CNODE1, CNODE2 and CMANAGER virtual machines:

1. LXFMT HELPCMS
2. LXFMT MODULE
3. SWAPGEN EXEC
4. SWAPGEN HELPCMS
5. PROFILE EXEC

9.1.1 SWAPGEN

Files 1-4 belong to the SWAPGEN package downloadable from:

<http://download.sinenomine.net/swapgen/>

Instructions for downloading and installing this package are available at that site.

9.1.2 PROFILE EXEC

The PROFILE EXEC is the first script executed when a virtual machine is logged on. It is used to set up the environment before booting Linux:

Figure 37: Sample PROFILE EXEC

```
/* REXX */
'CP SET PF12 RETRIEVE FORWARD'
'CP SET PF24 RETRIEVE BACKWARD'
'VMLINK TCPIP 592'
'SWAPGEN 152 200000 (FBA'
Say 'Enter a non-blank character and ENTER (or two ENTERs) within 10'
Say ' seconds to interrupt Linux IPL.'
'WAKEUP +00:10 (CONS'
If rc = 6 Then Do
    Say 'Interrupt: entering CMS.'
    Pull /* Clear Stack */
    Say 'IPL 150 CLEAR to boot Linux from DASD'
End
Else Do
    'CP IPL 150 CLEAR'
End
```

9.2 SELinux Policy

The `snahao.pp` file contains SELinux definitions that permit the operation of the fence agents. It is built from the `snahao.te` file:

Figure 38: SELinux Definitions

```
module snahao 1.0;

require {
    type var_log_t;
    type corosync_t;
    type qdiskd_t;
    type user_tmpfs_t;
    class unix_dgram_socket sendto;
    class file { read getattr };
}

#===== corosync_t =====
allow corosync_t self:unix_dgram_socket sendto;

#===== qdiskd_t =====
allow qdiskd_t user_tmpfs_t:file getattr;
allow qdiskd_t var_log_t:file read;

require {
    type kernel_t;
    type fenced_t;
    type port_t;
    class tcp_socket name_connect;
    class system module_request;
    class socket create;
}

#===== fenced_t =====

allow fenced_t kernel_t:system module_request;
allow fenced_t port_t:tcp_socket name_connect;
allow fenced_t self:socket create;

require {
    type file_t;
    type httpd_t;
    class dir { search getattr };
    class file { read getattr open execute };
}

#===== gfs/httpd_t =====
allow httpd_t file_t:dir { getattr search };
allow httpd_t file_t:file { read getattr open execute };
```

If you need to regenerate the `.pp` file then follow these steps:

1. Create .mod file:
checkmodule -M -m -o snahao.mod snahao.te
2. Create .pp file:
semodule_package -o snahao.pp -m snahao.mod

To load the policy, issue the following command:
semodule -i snahao.pp

9.3 SMAPI Test – RECYCLE EXEC

This test program uses the RXSOCKET package available on the system 'S' disk. Note, please replace the value of the "AuthPass" variable below with the password chosen for the userid CMANAGER. This EXEC will attempt to recycle the user CMANAGER, although you can choose any userid you prefer. A log "RECYCLE LOG" is written that logs requests and responses (return and reason codes, and text).

Figure 39: SMAPI Test - RECYCLE EXEC

```
/* RECYCLE EXEC */
'PIPE cms ERASE RECYCLE LOG | hole'
parse upper arg RecycleUser
TCPPort = 44444
Logging = 1
E_STAT = 1; E_INIT = 2; E_CONN = 3
E_PROT = 4; E_SIGN = 5; E_CLOSE= 6
E_SEND = 7; E_READ = 8
AuthUser = 'CMANAGER'
AuthPass = 'XXXXXXXX'
AuthTrgt = 'CMANAGER'
parse value SOCKET('INITIALIZ','DSC') with Rc ErrName ErrMsg
call RECYCLE_IMAGE
exit

CONNECT:
  parse value SOCKET('SOCKET') with Rc DSCSD
  parse value SOCKET('SETSOCKOPT',DSCSD,'Sol_Socket','SO_REUSEADDR','ON') ,
    with Rc Rest
  MyPort = RANDOM(50000,60000)
  parse value SOCKET('BIND',DSCSD,'AF_INET' MyPort 'LOCALHOST') ,
    with Rc Rest
  parse value SOCKET('CONNECT',DSCSD,'AF_INET' TCPPort 'LOCALHOST') ,
    with Rc Rest
return

XLATE:

  parse arg Value,Dir

  'PIPE var Value | xlate' Dir '|' var NewValue

return NewValue
```

RECYCLE_IMAGE:

```

call CONNECT
SMAPI_Fn = XLATE('Image_Recycle','e2a')
AuthUser = XLATE(STRIP(AuthUser),'e2a')
AuthPass = XLATE(STRIP(AuthPass),'e2a')
AuthTrgt = XLATE(STRIP(AuthTrgt),'e2a')
LFn      = D2C(LENGTH(SMAPI_Fn),4)
LUser    = D2C(LENGTH(AuthUser),4)
LPass    = D2C(LENGTH(AuthPass),4)
Target   = XLATE(RecycleUser,'e2a')
LTarget  = D2C(LENGTH(Target),4)
NoLog.0  = 0
ReqBody  = LFn || SMAPI_Fn ,
           || LUser || AuthUser ,
           || LPass || AuthPass ,
           || LTarget || Target
Req      = D2C(LENGTH(ReqBody),4) || ReqBody
Rc       = SEND_REQUEST(DSCSD,Req)
if (Rc = 0) then
do
  Response = GET_RESPONSE(DSCSD)
  if (Response = '') then
  do
    SockRc = Rc
    ErrSock = DSCSD
    Rc      = 1
    Reason  = E_READ
  end
  else
  do
    Rc = CLOSE_SOCKET()
    parse var Response 1 Req +4 Rc +4 Reason +4 LArray +4 Array
    Rc = C2D(Rc)
    Reason = C2D(Reason)
    if (Rc = 0) then
    do
      LogMsg = 'Successfully Recycled User ('Rc','Reason')'
      call LOGIT 'L',Rc LogMsg
    end
    else
    do
      LogMsg = 'Error retrieving NOLOG entries ('Rc','Reason')'
      call LOGIT 'L',Rc LogMsg
    end
    NoLog.0 = I_NoLog
  end
end
else
  Rc = 0
end
return Rc

```

SEND_REQUEST:

```

parse arg SD,Request
parse value SOCKET('SEND',SD,Request) with Rc ErrName ErrMsg
if (Rc <> 0) then
do
    SockRc = Rc
    ErrSock = DSCSD
    Rc = 1
    Reason = E_SEND
end
else
do
    call LOGIT '>',Rc Request
    parse value SOCKET('RECV',SD,4) with Rc Size ReqId
    if (Rc <> 0) then
    do
        SockRc = Rc
        ErrSock = SD
        Rc = 1
        Reason = E_READ
    end
    call LOGIT '<',Rc ReqId
end

return Rc

GET_RESPONSE:

parse arg SD
Data = ''
parse value SOCKET('RECV',SD,4) with Rc Size Len
if (Rc = 0) then
do
    Len = C2D(Len)
    if (Len > 0) then
    do
        do while Len > 0
            parse value SOCKET('RECV',SD,Len) with Rc Size Msg
            if (Rc = 0) then
                Data = Data || Msg
            else
                return ''
            Len = Len - Size
        end
    end
end
call LOGIT '<',Rc Data

return Data

LOGIT:

parse arg LogType,Rc LogData
if (Logging) then
do
    RetC = Rc

```

```

select
  when LogType = 'L' then
  do
    'PIPE (name LOGMSG)',
    '| var LogData',
    "| literal" DATE('S') TIME('L') LogType Rc,
    '| >> RECYCLE LOG A'
  end
  when LogType = '<' | LogType = '>' then
  do
    'PIPE (name LOGIT end ?)',
    '| var LogData',
    '| fblock 32',
    '| spec 1-* c2x 1 /*/ 70 1-* n',
    '| xlate 71-* a2e',
    '| xlate 71-* 00-3f 4b',
    '| spec /+/ 1 recno from 0 by 32 2.4 1-* 8 /*/ n',
    "| literal" DATE('S') TIME('L') LogType Rc,
    '| >> RECYCLE LOG A'
  end
  otherwise
end
'FINIS RECYCLE LOG A'
Rc = RetC
end

return

CLOSE_SOCKET:

  parse value SOCKET('CLOSE', DSCSD) with Rc .
  if (Rc <> 0) then
  do
    Reason  = E_CLOS
    SockRc  = Rc
    Rc      = 1
    ErrSock = DSCSD
  end

return Rc

```

10 Additional Resources

10.1 HAO Documentation

10.2 IBM z/VM Documentation

1. [z/VM Systems Management Application Programming](#)
2. [z/VM 6.2 General Information](#)

10.3 Red Hat Documentation

3. [RHEL6 High-Availability Add-On Overview](#)
4. [RHEL6 High-Availability Add-On Cluster Administration](#)
5. [RHEL6 Installation](#)

10.4 Other Documentation

6. [Wikipedia – VM Operating System](#)
7. [Wikipedia – High Availability Clustering](#)

