Cluster Installation using SNA HAO for RHEL 7

Document ID: HAOZ-002

February 27, 2015

Document History

Date	*Revision/ Reissue*	*Nature of Change*
Oct 1, 2014	Version 1	New Release

Table of Contents

List of Tables

List of Figures

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 7 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 pcs command line 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 (for example, GFS2 volumes) when using ECKD devices.

As an additional exercise the installation and configuration of two other clustering components is demonstrated:

1. gluster - GlusterFS is an open source, distributed file system capable of scaling to several petabytes (actually, 72 brontobytes!) and handling thousands of clients. GlusterFS clusters together storage building blocks over Infiniband RDMA or TCP/IP interconnect, aggregating disk and memory resources and managing data in a single global namespace. GlusterFS is based on a stackable user space design and can deliver exceptional performance for diverse workloads. [10]
2. DRBD - DRBD refers to block devices designed as a building block to form high availability (HA) clusters. Mirroring a whole block device via an assigned network does this. DRBD can be understood as network based raid-1. [11]

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 zPDT.

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 SSI.
- Red Hat Enterprise Linux (RHEL) 7.x for System z. The example cluster in this document was implemented using RHEL version 7.0
- A z/VM directory manager such as CA VM:Secure or IBM DIRMAINT (see topic 2.3.2 " Directory Management Applications Recommendation" on page 2-6).
- The SNA HAO for RHEL on System z "yum" repository. Obtaining access to this repository is described in section **Error! Reference source not found.** "**Error! Reference source not found.**" on page **Error! Bookmark not defined.** 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 11.2 "IBM z/VM Documentation" on page 27.

1.3.4 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.

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." [8]

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
pacemaker	Pacemaker is an advanced, scalable High-Availability cluster resource manager for Corosync, CMAN and/or Linux-HA. It supports more than 16 node clusters with significant capabilities for managing resources and dependencies. It will run scripts at initialization, when machines go up or down, when related resources fail and can be configured to periodically check resource health. It consists of several services: • cib – Cluster configuration • stonithd – "Shoot the other node in the head" daemon: cluster protection • lrmd – Local resource measurement • attrd – Cluster configuration co-ordination • pengine – Policy engine • crmd – Cluster resource management
pcs	pcs is a corosync and pacemaker configuration tool. It permits users to easily view, modify and created pacemaker based clusters.
resource-agents	A set of scripts to interface with several services to operate in a High Availability environment for both Pacemaker and rgmanager service managers.
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.
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.
dlm_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 dlm_controld to receive these events and reconfigure the kernel dlm as needed. dlm_controld

Service	Description
	controls and configures the dlm through sysfs and configfs files that are considered dlm-internal interfaces.
corosync	Corosync provides clustering infrastructure such as membership, messaging and quorum. It is a group communication system with additional features for implementing 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. The services provided by the Corosync Service Engine internal API are: • An implementation of the Totem Single Ring Ordering and Membership protocol providing the Extended Virtual Synchrony model for messaging and membership. • The coroipc 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. Additionally Corosync provides several default service engines: • 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
libqb	libqb provides high performance client server reusable features. Initially these are IPC and poll.

2.1.3 Configuration

Configuration is performed using either the pcs command line toolset or via a browser interface provided by the pacemaker daemon (pcsd). See “10 pcs Command Reference” on page 7. The corosync daemon defines basic cluster operation. The following definitions show some of the statements used in this file.

Figure 1: Example of XML-based Corosync Definition File

```
totem {
  version: 2
  secauth: off
  cluster_name: rh7cluster
  transport: udpu
}

nodelist {
  node {
        ring0_addr: rh7cn1.sinenomine.net
        nodeid: 1
       }
  node {
        ring0_addr: rh7cn2.sinenomine.net
        nodeid: 2
       }
}

quorum {
  provider: corosync_votequorum
  two_node: 1
}

logging {
  to_syslog: yes
}
```

The resources and constraints that are created in the various tasks within this document are stored in another XML file: `/var/lib/pacemaker/cib/cib.xml`. It contains statements like the ones shown below:

Figure 2: Example of XML-based Resource Definition File

```
<cib admin_epoch="0" cib-last-written="Tue Oct 28 16:45:30 2014"
crm_feature_set="3.0.7" epoch="501" have-quorum="1" num_updates="0" update-
client="cibadmin" update-origin="rh7cn1.sinenomine.net" validate-with="pacemaker-1.2"
dc-uuid="1">
  <configuration>
    <crm_config>
      <cluster_property_set id="cib-bootstrap-options">
        <nvpair id="cib-bootstrap-options-dc-version" name="dc-version" value="1.1.10-
29.el7-368c726"/>
        <nvpair id="cib-bootstrap-options-cluster-infrastructure" name="cluster-
infrastructure" value="corosync"/>
        <nvpair id="cib-bootstrap-options-no-quorum-policy" name="no-quorum-policy"
value="freeze"/>
      </cluster_property_set>
    </crm_config>
    <nodes>
      <node id="1" uname="rh7cn1.sinenomine.net">
```

```
        <instance_attributes id="nodes-1"/>
      </node>
      <node id="2" uname="rh7cn2.sinenomine.net">
        <instance_attributes id="nodes-2"/>
      </node>
    </nodes>
    <resources>
      <primitive class="stonith" id="ZVMPOWER" type="fence_zvm">
        <instance_attributes id="ZVMPOWER-instance_attributes">
          <nvpair id="ZVMPOWER-instance_attributes-ipaddr" name="ipaddr"
value="VSMREQIU"/>
          <nvpair id="ZVMPOWER-instance_attributes-pcmk_host_map" name="pcmk_host_map"
value="rh7cn1.sinenomine.net:RH7CN1;rh7cn2.sinenomine.net:RH7CN2"/>
          <nvpair id="ZVMPOWER-instance_attributes-pcmk_host_list"
name="pcmk_host_list" value="rh7cn1.sinenomine.net;rh7cn2.sinenomine.net"/>
          <nvpair id="ZVMPOWER-instance_attributes-pcmk_host_check"
name="pcmk_host_check" value="static-list"
/>
        </instance_attributes>
        <operations>
          <op id="ZVMPOWER-monitor-interval-60s" interval="60s" name="monitor"/>
        </operations>
      </primitive>
:
:
    </resources>
    <constraints>
      <rsc_order first="dlm-clone" first-action="start" id="order-dlm-clone-clvmd-
clone-mandatory" then="clvmd-clone" then-action="start"/>
      <rsc_colocation id="colocation-clvmd-clone-dlm-clone-INFINITY" rsc="clvmd-clone"
score="INFINITY" with-rsc="dlm-clone"/>
      <rsc_order first="clvmd-clone" first-action="start" id="order-clvmd-clone-
apachegroup-mandatory" then="apachegroup" then-action="start"/>
      <rsc_order first="clvmd-clone" first-action="start" id="order-clvmd-clone-
clusterfs-clone-mandatory" then="clusterfs-clone" then-action="start"/>
      <rsc_colocation id="colocation-clusterfs-clone-clvmd-clone-INFINITY"
rsc="clusterfs-clone" score="INFINITY" with-rsc="clvmd-clone"/>
      <rsc_colocation id="colocation-GDPSFS-GDPSClone-INFINITY" rsc="GDPSFS"
score="INFINITY" with-rsc="GDPSClone" with-rsc-role="Master"/>
      <rsc_order first="GDPSClone" first-action="promote" id="order-GDPSClone-GDPSFS-
mandatory" then="GDPSFS" then-action="start"/>
      <rsc_location id="location-brick1-rh7cn1.sinenomine.net-INFINITY"
node="rh7cn1.sinenomine.net" rsc="brick1" score="INFINITY"/>
      <rsc_location id="location-brick2-rh7cn2.sinenomine.net-INFINITY"
node="rh7cn2.sinenomine.net" rsc="brick2" score="INFINITY"/>
      <rsc_colocation id="colocation-brick1-brick2--INFINITY" rsc="brick1" score="-
INFINITY" with-rsc="brick2"/>
      <rsc_colocation id="colocation-gvol1-brick1-INFINITY" rsc="gvol1"
score="INFINITY" with-rsc="brick1"/>
      <rsc_colocation id="colocation-gvol2-brick2-INFINITY" rsc="gvol2"
score="INFINITY" with-rsc="brick2"/>
      <rsc_order first="glusterd-clone" first-action="start" id="order-glusterd-clone-
gvol1-mandatory" then="gvol1" then-action="start"/>
      <rsc_order first="glusterd-clone" first-action="start" id="order-glusterd-clone-
gvol2-mandatory" then="gvol2" then-action="start"/>
      <rsc_colocation id="colocation-store1-brick1-INFINITY" rsc="store1"
score="INFINITY" with-rsc="brick1"/>
      <rsc_colocation id="colocation-store2-brick2-INFINITY" rsc="store2"
score="INFINITY" with-rsc="brick2"/>
      <rsc_order first="gvol1" first-action="start" id="order-gvol1-store1-mandatory"
then="store1" then-action="start"/>
```

```
      <rsc_order first="gvol2" first-action="start" id="order-gvol2-store2-mandatory"
then="store2" then-action="start"/>
    </constraints>
  </configuration>
</cib>
```

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 7.0
- 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 as "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: CP; 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 stand-alone machine. A given VM mainframe typically runs hundreds or thousands of virtual machine instances. [7]

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 so far 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.

RH7CN1

Function

A node of the cluster.

Software Installed

- RHEL 7.0
- pcs
- fence-agents
- gfs2 kernel module

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

RH7CN2

Function

A node of the cluster.

Software Installed

- RHEL 7.0
- pcs
- fence-agents
- gfs2 kernel module

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

2.2.1.5 Storage Needed

The following table describes the minidisks used in the cluster configuration created in this documented.

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
	XFS – Apache server content	200	500 cyl
RH7CN1	RHEL 7.0 system	150-151	2 x 3338 cyl
	Swap – VDISK	152	200000 blocks
	gluster volume	300	200 cyl
	drbd volume	301	200 cyl
RH7CN2	RHEL 7.0 system	150-151	2 x 3338 cyl
	Swap – VDISK	152	200000 blocks
	gluster volume	300	200 cyl
	drbd volume	301	200 cyl

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 * CF0 CF0 SR
7. #CP CPACCESS PMAINT CF0 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.

2.3.3 Performance Management

The use of a Performance Management tool such as Velocity Software’s zVPS or IBM’s Performance Toolkit is also strongly recommended, as clustering requires communication between and coordination of multiple virtual machines. It is important to monitor and understand the performance profiles of the clustering components.

2.3.4 SSI Considerations

When running as a virtual machine in a member of an SSI cluster use the TCP/IP-based z/VM fence agent. See “8 z/VM Fence Devices” on page 8-9.

2.3.5 Firewall Rules

The HAO fence agent uses either TCP/IP or IUCV to perform the fencing operation. In addition, some of the clustering demons also require the use of certain IP ports. These details may be found at “3.10 Task: Install pcs and fence agents on Nodes” on page 3-6.

In addition, if you are using SELinux you will need to read “3.9 Task: Enabling SELinux Policy for Clustering [SELinux Only]” on page 3-6.

3 Installation of Infrastructure Components for a Sample 2 Node Cluster

3.1 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.

3.1.1 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.

3.1.2 Task List

#	Description	☒
Creating the z/VM Userid for Shared Cluster Disks		
1	Create user COMMON	
Creating the z/VM Userids for Cluster Operation		
1	Prepare the source for PROFILE EXEC	
2	Create user RH7CN1	
3	Prepare A disk of RH7CN1	
4	Create user RH7CN2	
5	Prepare A disk of RH7CN2	
1	Logon as a privileged user	
2	Access z/VM SMAPI Configuration minidisk/SFS directory	
3	Edit VSMWORK1 AUTHLIST	
4	Test VSMWORK1	

#	Description	☒

3.2 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.

3.2.1 Task: Create User COMMON

Create a file COMMON DIRECT A containing the following:

Figure 3: COMMON z/VM Virtual Machine Definition

```
USER COMMON NOLOG 8K 8K
  MDOPT FORMAT LABEL GFS001
  MDISK 153 3390 * 3338 *
  MINIOPT NOMDC
  MDOPT FORMAT LABEL APC001
  MDISK 200 3390 * 0500 *
  MINIOPT NOMDC
```

Issue the following command:

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

3.3 Creating the z/VM Userids for the Cluster Userids

3.3.1 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 1. This will be used by VMSECURE when creating the virtual machines.

3.3.2 Task: Create User RH7CN1

Create a file RH7CN1 DIRECT A containing the following, specifying a password that meets your installation standards:

Figure 4: RH7CN1 z/VM Virtual Machine Definition

```
USER RH7CN1 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 AUTOCR
  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 MW
  NICDEF C600 TYPE QDIO DEVICES 3
  MDOPT FORMAT
  MDISK 150 3390 * 3338 * M
  MDOPT FORMAT
  MDISK 151 3390 * 3338 * M
  MDOPT FORMAT
  MDISK 300 3390 * 200 * M
  MDOPT FORMAT
  MDISK 301 3390 * 200 * M
```

Issue the following command:

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

3.3.3 Task: Prepare 'A' Disk of RH7CN1

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

3.3.4 Task: Create User RH7CN2

Create a file `RH7CN2 DIRECT A` containing the following, specifying a password that meets your installation standards:

Figure 5: RH7CN2 z/VM Virtual Machine Definition

```
USER RH7CN2 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 AUTOCR
  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 MW
NICDEF C600 TYPE QDIO DEVICES 3
MDOPT FORMAT
MDISK 150 3390 * 3338 * M
MDOPT FORMAT
MDISK 151 3390 * 3338 * M
MDOPT FORMAT
MDISK 300 3390 * 200 * M
MDOPT FORMAT
MDISK 301 3390 * 200 * M
```

Issue the following command:

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

3.3.5 Task: Prepare 'A' Disk of RH7CN2

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

3.4 Enabling the z/VM SMAPI Server

3.4.1 Task: Log On As a Privileged User

LOGON MAINT

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

VMLINK .DIR VMSYS:VSMWORK1. (WRITE

3.4.3 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:

Figure 6: VSMWORK1 AUTHLIST File

```
Column 1            Column 66              Column 131
↓                   ↓                      ↓
```

```
RH7CN1                  ALL                     IMAGE_OPERATIONS
RH7CN2                  ALL                     IMAGE_OPERATIONS
```

3.4.4 Task: Test VSMWORK1 (optional)

Use the RECYCLE EXEC described at "9.3 SMAPI Test – RECYCLE EXEC" on page 3, to test the SMAPI configuration.

3.5 Installing RHEL for System z on RH7CN1 and RH7CN2

Follow your installation's recommended procedures to install Red Hat Enterprise Linux 7.x on the private disk storage for RH7CN1 and RH7CN2.

Note: Only use disks 0150 and 0151 as installation targets, 0153 and 0200 will be cluster resources.

3.5.1 Task: Run "yum update" on RH7CN1

3.5.2 Task: Run "yum update" on RH7CN2

3.6 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.

3.7 Configuring and Enabling the SNA HAO Repository on Cluster Nodes

The download materials include the following important files and directories.

Table 3: HAO Materials

3.10.0-123.*x*	Contains the gfs2 kernel module for 3.10.0-123.*x* kernels
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.

3.8 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.

3.8.1 Task: Install GFS2 Kernel Module on RH7CN1 and RH7CN2

```
yum install
https://USERID:PASSWORD@files.sinenomine.net/hao/cluster/3.0.10-
123.8.1/gfs2-kmod-3.10.0-123.8.1.cl7.s390x.rpm
depmod –a
modprobe gfs2
```

3.9 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
```

3.10 Task: Install pcs and fence agents on Nodes

On each node in the cluster, install the High Availability Add-On software packages along with all available fence agents from the repository:

```
# yum install pcs fence-agents-all
```

3.11 Task: Enabling Network Ports [Firewall Only]

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

```
# firewall-cmd --permanent --add-service=high-availability
# firewall-cmd --add-service=high-availability
```

3.12 Task: Enabling the Cluster Administration ID on Nodes

The pcs tool requires use of the `hacluster` account to communicate between nodes and update the configuration. On each node, set the password for the `hacluster` account:

```
# passwd hacluster
Changing password for user hacluster.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
```

3.13 Task: Enable the pcs service

On both nodes:

```
# systemctl start pcsd.service
# systemctl enable pcsd.service
```

3.14 Task: Authorize hacluster to administer Cluster

On one of the nodes:

```
# pcs cluster auth RH7CN1.sinenomine.net RH7CN2.sinenomine.net
Username: hacluster
Password:
RH7CN1.sinenomine.net: Authorized
RH7CN2.sinenomine.net: Authorized
```

3.15 Task: Create the Cluster

On one of nodes:

```
# pcs cluster setup --start --name rh7cluster rh7cn1.sinenomine.net
rh7cn2.sinenomine.net
rh7cn1.sinenomine.net: Succeeded
rh7cn1.sinenomine.net: Starting Cluster...
rh7cn2.sinenomine.net: Succeeded
rh7cn2.sinenomine.net: Starting Cluster...
```

3.16 Task: Enable Cluster to start on boot

On one of the nodes:

```
# pcs cluster enable --all
```

3.17 Task: Check the Cluster Status

On one of the nodes:

```
# pcs cluster status
Cluster Status:
 Last updated: Wed Oct  1 17:37:30 2014
 Last change: Wed Oct  1 16:07:59 2014 via crmd on rh7cn1.sinenomine.net
 Stack: corosync
 Current DC: rh7cn1.sinenomine.net (1) - partition with quorum
 Version: 1.1.10-29.el7-368c726
 2 Nodes configured
 0 Resources configured

PCSD Status:
  rh7cn1.sinenomine.net: Online
  rh7cn2.sinenomine.net: Online
```

3.18 Task: Install LVM clustering and GFS2 Utilities

On both nodes:

```
# yum install lvm2-cluster gfs2-utils
```

3.19 Task: Configure Fencing on Nodes

On one of the nodes, examine the parameters required by the fencing agent "fence_zvm" and note the parameters that are marked as "required":

```
# pcs stonith describe fence_zvm
Stonith options for: fence_zvm
  port (required): Name of the Virtual Machine to be fenced
  ipaddr (required): Name of the SMAPI IUCV Server Virtual Machine
  zvmsys: Node of the SMAPI IUCV Server Virtual Machine
  action: Fencing action
  delay: Time to delay fencing action in seconds
  usage: Print usage
  stonith-timeout: How long to wait for the STONITH action to complete per a
stonith device.
  priority: The priority of the stonith resource. Devices are tried in order
of highest priority to lowest.
  pcmk_host_map: A mapping of host names to ports numbers for devices that do
not support host names.
  pcmk_host_list: A list of machines controlled by this device (Optional
unless pcmk_host_check=static-list).
  pcmk_host_check: How to determine which machines are controlled by the
device.
```

On one of the nodes enter the following command which will specify that the fence_zvm fencing agent is to be used, associate the virtual machine name with the hostname, and provide a list of hosts that are under the control of this fencing agent instance:

```
# pcs stonith create ZVMPOWER fence_zvm params ipaddr="VSMREQIU"
pcmk_host_map="rh7cn1.sinenomine.net:RH7CN1;rh7cn2.sinenomine.net:RH7CN
2" pcmk_host_list="rh7cn1.sinenomine.net;rh7cn2.sinenomine.net"
pcmk_host_check=static-list
Warning: missing required option(s): 'port' for resource type:
stonith:fence_zvm
```

You may ignore the warning message.

Verify the fence agent was correctly configured:

```
# pcs stonith show ZVMPOWER
 Resource: ZVMPOWER (class=stonith type=fence_zvm)
  Attributes: ipaddr=VSMREQIU
pcmk_host_map=rh7cn1.sinenomine.net:RH7CN1;rh7cn2.sinenomine.net:RH7CN2
pcmk_host_list=rh7cn1.sinenomine.net;rh7cn2.sinenomine.net
pcmk_host_check=static-list
  Operations: monitor interval=60s (ZVMPOWER-monitor-interval-60s)
```

3.20 Task: Testing Fencing

With both nodes up and running, test the fencing operation by placing RH7CN1 into a non-responsive state and have RH7CN2 invoke the fence agent to cause RH7CN1 to be forced off the system and logged back on.

Logon to the user RH7CN1 via TN3270 and then enter #CP SLEEP on the console. This will stop any Linux activity and RH7CN2 will soon detect RH7CN1 being unresponsive.

On RH7CN2, the fence daemon log file (/var/log/cluster/fenced.log) will contain entries similar to these:

Figure 7: Fencing - System Log Messages

```
Oct  5 09:45:26 rh7cn2 stonith-ng[1759]: notice: can_fence_host_with_device:
ZVMPOWER can not fence 2: static-list
Oct  5 09:45:26 rh7cn2 stonith-ng[1759]: notice: remote_op_done: Operation
reboot of 2 by rh7cn1.sinenomine.net for stonith-
api.7257@rh7cn1.sinenomine.net.55670f2a: No such device
Oct  5 09:45:26 rh7cn2 crmd[1764]: notice: tengine_stonith_notify: Peer 2 was
not terminated (reboot) by rh7cn1.sinenomine.net for rh7cn1.sinenomine.net: No
such device (ref=55670f2a-3b06-44af-9799-f4acb0eb3d41) by client stonith-
api.7257
Oct  5 09:45:44 rh7cn2 dlm_controld[2910]: 43695 fence wait 1 pid 11991
running
Oct  5 09:45:46 rh7cn2 dlm_controld[2910]: 43697 vol1 wait for fencing
Oct  5 09:45:46 rh7cn2 dlm_controld[2910]: 43697 clvmd wait for fencing
```

```
Oct  5 09:46:01 rh7cn2 stonith-ng[1759]: notice: can_fence_host_with_device:
ZVMPOWER can fence rh7cn1.sinenomine.net (aka. 'RH7CN1'): static-list
Oct  5 09:47:13 rh7cn2 stonith-ng[1759]: notice: can_fence_host_with_device:
ZVMPOWER can fence rh7cn1.sinenomine.net (aka. 'RH7CN1'): static-list
Oct  5 09:47:13 rh7cn2 kernel: NET: Registered protocol family 32
Oct  5 09:47:15 rh7cn2 corosync[1685]: [TOTEM ] A processor failed, forming
new configuration.
Oct  5 09:47:17 rh7cn2 fence_zvm[12644]: Recycling of RH7CN1 successful
Oct  5 09:47:17 rh7cn2 stonith-ng[1759]: notice: log_operation: Operation
'reboot' [12644] (call 2 from crmd.1907) for host 'rh7cn1.sinenomine.net' with
device 'ZVMPOWER' returned: 0 (OK)
Oct  5 09:47:17 rh7cn2 corosync[1685]: [TOTEM ] A new membership
(172.17.16.148:436) was formed. Members left: 1
Oct  5 09:47:17 rh7cn2 crmd[1764]: notice: peer_update_callback: Our peer on
the DC is dead
Oct  5 09:47:17 rh7cn2 crmd[1764]: notice: do_state_transition: State
transition S_NOT_DC -> S_ELECTION [ input=I_ELECTION
cause=C_CRMD_STATUS_CALLBACK origin=peer_update_callback ]
Oct  5 09:47:17 rh7cn2 corosync[1685]: [QUORUM] Members[1]: 2
Oct  5 09:47:17 rh7cn2 crmd[1764]: notice: crm_update_peer_state:
pcmk_quorum_notification: Node rh7cn1.sinenomine.net[1] - state is now lost
(was member)
Oct  5 09:47:17 rh7cn2 kernel: dlm: closing connection to node 1
Oct  5 09:47:17 rh7cn2 pacemakerd[1706]: notice: crm_update_peer_state:
pcmk_quorum_notification: Node rh7cn1.sinenomine.net[1] - state is now lost
(was member)
Oct  5 09:47:17 rh7cn2 corosync[1685]: [MAIN  ] Completed service
synchronization, ready to provide service.
Oct  5 09:47:17 rh7cn2 stonith-ng[1759]: notice: remote_op_done: Operation
reboot of rh7cn1.sinenomine.net by rh7cn2.sinenomine.net for
crmd.1907@rh7cn1.sinenomine.net.129f81f0: OK
Oct  5 09:47:17 rh7cn2 crmd[1764]: notice: tengine_stonith_notify: Peer
rh7cn1.sinenomine.net was terminated (reboot) by rh7cn2.sinenomine.net for
rh7cn1.sinenomine.net: OK (ref=129f81f0-4aa2-430b-b408-dca802134a92) by client
crmd.1907
Oct  5 09:47:17 rh7cn2 crmd[1764]: notice: tengine_stonith_notify: Target may
have been our leader rh7cn1.sinenomine.net (recorded: <unset>)
Oct  5 09:47:17 rh7cn2 crmd[1764]: notice: do_state_transition: State
transition S_ELECTION -> S_INTEGRATION [ input=I_ELECTION_DC
cause=C_FSA_INTERNAL origin=do_election_check ]
Oct  5 09:47:17 rh7cn2 crmd[1764]: notice: do_dc_takeover: Marking
rh7cn1.sinenomine.net, target of a previous stonith action, as clean
```

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

```
09:47:17 OPERATOR *3 GRAF L0003 LOGOFF AS  RH7CN1   USERS = 59    FORCED BY VSMWORK1
09:47:17 VSMWORK1 *8 09:47:36 GRAF L0003 LOGOFF AS  RH7CN1   USERS = 59    FORCED BY
VSMWORK1
09:47:18 OPERATOR *3 AUTO LOGON  ***       RH7CN1   USERS = 60    BY VSMWORK1
```

RH7CN1 will now reboot and automatically rejoin the cluster.

4 Configuring Cluster Resources and Services

4.1 Adding Shared GFS2 Disk

To add a GFS2 disk that is shared between nodes, following the following steps.

4.1.1 Task: Activate Shared Cluster Resource Disks on RH7CN1 and RH7CN2

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
# chccwdev –e 0.0.0200
```

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

```
-  parameters="root=/dev/mapper/rhel_rh7cn1-root cio_ignore=all,!condev
rd.dasd=0.0.0151 rd.dasd=0.0.0150 rd.dasd=0.0.0152
rd.lvm.lv=rhel_rh7cn1/boot vconsole.keymap=us
rd.lvm.lv=rhel_rh7cn1/root crashkernel=auto  vconsole.font=latarcyrheb-
sun16 rd.lvm.lv=rhel_rh7cn1/swap LANG=en_US.UTF-8"

+  parameters="root=/dev/mapper/rhel_rh7cn1-root cio_ignore=all,!condev
rd.dasd=0.0.0151 rd.dasd=0.0.0150 rd.dasd=0.0.0152 rd.dasd=0.0.0153
rd.dasd=0.0.0200 rd.dasd=0.0.0300 rd.dasd=0.0.0301
rd.lvm.lv=rhel_rh7cn1/boot vconsole.keymap=us
rd.lvm.lv=rhel_rh7cn1/root crashkernel=auto  vconsole.font=latarcyrheb-
sun16 rd.lvm.lv=rhel_rh7cn1/swap LANG=en_US.UTF-8"
```

Run `zipl` to make the changes permanent:

```
# zipl
```

4.1.2 Task: Format and Partition Shared Cluster Disks on RH7CN1

On RH7CN1 Format and partition the two shared disks (gfs2 and the apache volume):

```
# dasdfmt –b 4096 –y /dev/dasdd
# fdasd -a /dev/dasdd
# dasdfmt –b 4096 –y /dev/dasde
# fdasd –a /dev/dasde
```

4.1.3 Task: Enable Logical Volume Clustering

To use clustered logical volumes, two settings need to be set or confirmed in the lvm configuration file. The `lvmconf` command will perform this task. On both nodes:

```
# lvmconf --enable-cluster
# systemctl reboot
```

4.1.4 Task: Set Cluster Property for GFS2

When the nodes have rebooted, on one of the nodes:

```
# pcs property set no-quorum-policy=freeze
```

Command explanation: When quorum is lost the remaining partition will do nothing until quorum is regained – GFS2 requires quorum to operate.

4.1.5 Task: Create a DLM resource

The distributed lock manager (DLM) is required for GFS2 and clustered logical volume operation so on one of the nodes create a DLM resource:

```
# pcs resource create dlm ocf:pacemaker:controld op monitor
interval=30s on-fail=fence clone interleave=true ordered=true
```

Command explanation: Create the distributed lock manager resource that will run on all nodes in the cluster.

4.1.6 Task: Create CLVM Resource

On one of the nodes:

```
# pcs resource create clvmd ocf:heartbeat:clvm op monitor interval=30s
on-fail=fence clone interleave=true ordered=true
```

Command explanation: This pcs command creates a clustered logical volume daemon resource named “clvmd”, which will run on all nodes.

4.1.7 Task: Define Dependency between DLM and CLVMD

On one of the nodes:

```
# pcs constraint order start dlm-clone then clvmd-clone
# pcs constraint colocation add clvmd-clone with dlm-clone
```

Command explanation: These commands define the dependency that the clvmd resource has on dlm and that both resources need to live on the same nodes.

4.1.8 Task: Create Physical Volume

```
pvcreate /dev/dasdd1
```

4.1.9 Task: Create Cluster Volume Group

```
# vgcreate –Ay –cy vg_cluster /dev/dasdd1
```

4.1.10 Task: Create Logical Volume

```
# lvcreate –L 500m –n ha_lv vg_cluster
```

4.1.11 Task: Create GFS2 Filesystem

```
# mkfs.gfs2 -j 2 -r 32 -t rh7cluster: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.gfs2 -j 2 -J 16 -r 32 -t rh7cluster:vol1 /dev/mapper/vg_cluster-ha_lv
```

4.1.12 Task: Create a Cluster File System Resource

On both nodes:

```
# mkdir /mnt/gfs2-demo
```

On one of the nodes:

```
# pcs resource create clusterfs Filesystem
device="/dev/vg_cluster/ha_lv" directory="/mnt/gfs2-demo" fstype="gfs2"
“options=noatime" op monitor interval=10s on-fail=fence clone
interleave=true
```

Command explanation: This command defines the clusterfs resource that is a gfs2 file system residing on `/dev/vg_cluster/ha_lv` and will be mounted at `/mnt/gfs2-demo` on both nodes.

4.1.13 Task: Define Dependency between CLVMD and GFS2

On one of the nodes:

```
# pcs constraint order start clvmd-clone then clusterfs-clone
# pcs constraint colocation add clusterfs-clone with clvmd-clone
```

Command explanation: These commands ensure that the clusterfs resource is started after the cluster logical volume resource and that both resources reside on the nodes.

4.2 Creating HA Web Server on RH7CN1 and RH7CN2

4.2.1 Task: Install Apache

On both nodes:

```
# yum install –y httpd wget
```

4.2.2 Task: Update Apache Configuration

On both nodes, create the file `/etc/httpd/conf.d/cluster.conf` with the following contents:

```
<Location /server-status>
   SetHandler server-status
   Order deny,allow
   Deny from all
   Allow from 127.0.0.1
 </Location>
```

4.3 Create Cluster Resources

4.3.1 Task: Create Physical Volume for Apache

On one of the nodes:

```
# pvcreate /dev/dasde1
```

4.3.2 Task: Create Volume Group for Apache

On one of the nodes:

```
# vgcreate vg_apache /dev/dasde1
```

4.3.3 Task: Create Logical Volume for Apache

On one of the nodes:

```
# lvcreate -L300M -n apache_lv vg_apache
```

4.3.4 Task: Create xfs File System on Shared Disk

On one of the nodes:

```
# mkfs.xfs /dev/mapper/vg_apache-apache_lv
```

4.3.5 Task: Save Existing /var/www

On one node take a copy of the existing /var/www directory:

```
# tar –cf /tmp/www.tar /var/www
```

4.3.6 Task: Create a simple home page

On one of the nodes prepare the gfs2 file system with the content required by apache; create a test home page; ensure selinux settings are correct; and unmount the file system:

```
# mount /dev/mapper/vg_apache-apache_lv /var/www

# cd /

#  tar –xf /tmp/www.tar

# rm –f /tmp/www.tar

# cat <<-END >/var/www/html/index.html
<html>
 <title>Cluster Web Resource Home Page</title>
 <h1>Welcome</h1>
 <body>Hello</body>
</html>
END

# restorecon –R /var/www

# umount /var/www
```

4.3.7 Task: Create LVM Resource

On one of the nodes:

```
# pcs resource create httplvm LVM volgrpname=vg_apache exclusive=true --group apachegroup
```

Command explanation: Define a logical volume manager resource that consists of the vg_apache logical volume group defined earlier and make it a member of a resource group we call apachegroup.

4.3.8 Task: Create File System Resource

On one of the nodes:

```
# pcs resource create http_fs Filesystem device="/dev/mapper/vg_apache-apache_lv" directory="/var/www" fstype="xfs" --group apachegroup
```

Command explanation: Define an XFS file system resource resource that consists of the vg_apache-apache_lv logical volume defined earlier, set its mount point as /var/www, and make it a member of a resource group we call apachegroup.

4.3.9 Task: Create IP Address Resource

On one of the nodes, 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.

```
# pcs resource create VirtualIP IPaddr2 ip=172.17.16.3 cidr_netmask=24
--group apachegroup
```

Command explanation: Define a virtual IP address resource with an address of 172.17.16.3/24, and make it a member of a resource group we call apachegroup.

4.3.10 Task: Create Web Resource

On one of the nodes:

```
# pcs resource create Website apache
configfile="/etc/httpd/conf/httpd.conf"
statusurl="http://127.0.0.1/server-status" --group apachegroup
```

Command explanation: Define a website resource we call apache, specify its configuration file, define the server status URL (that we defined in /etc/httpd/conf.d/cluster.conf), and make it a member of a resource group we call apachegroup.

4.3.11 Task: Enable http Firewall Service

On both nodes:

```
# firewall-cmd --permanent --add-service=http

# firewall-cmd --add-service=http
```

Command explanation: These commands will enable http access through the firewall. The first command makes the service permanent. The second enables the service for this session.

4.3.12 Task: Define Startup Constraint

On one of the nodes:

```
# pcs constraint order start clvmd-clone then apachegroup
```

Command explanation: Ensure that DLM and CLVMD are running before starting the apachegroup resources.

4.3.13 Task: Verify Resources are Active

On one of the nodes, check the status of the cluster and resources using the `pcs status` command. The output should look similar to this:

Figure 8: Configuration status after Apache resource definition

```
# pcs status
Cluster name: rh7cluster
Last updated: Fri Oct  3 14:21:44 2014
Last change: Fri Oct  3 14:12:52 2014 via cibadmin on
rh7cn1.sinenomine.net
Stack: corosync
Current DC: rh7cn1.sinenomine.net (1) - partition with quorum
Version: 1.1.10-29.el7-368c726
2 Nodes configured
11 Resources configured


Online: [ rh7cn1.sinenomine.net rh7cn2.sinenomine.net ]

Full list of resources:

 ZVMPOWER   (stonith:fence_zvm): Started rh7cn1.sinenomine.net
 Clone Set: dlm-clone [dlm]
     Started: [ rh7cn1.sinenomine.net ]
     Stopped: [ rh7cn2.sinenomine.net ]
 Clone Set: clvmd-clone [clvmd]
     Started: [ rh7cn1.sinenomine.net ]
     Stopped: [ rh7cn2.sinenomine.net ]
 Clone Set: clusterfs-clone [clusterfs]
     Started: [ rh7cn1.sinenomine.net ]
     Stopped: [ rh7cn2.sinenomine.net ]
 Resource Group: apachegroup
     VirtualIP (ocf::heartbeat:IPaddr2):  Started rh7cn1.sinenomine.net
     Website   (ocf::heartbeat:apache):   Started rh7cn1.sinenomine.net
     httplvm   (ocf::heartbeat:LVM):   Started rh7cn1.sinenomine.net
     http_fs   (ocf::heartbeat:Filesystem):  Started
rh7cn1.sinenomine.net

PCSD Status:
  rh7cn1.sinenomine.net: Online
  rh7cn2.sinenomine.net: Online

Daemon Status:
  corosync: active/enabled
  pacemaker: active/enabled
  pcsd: active/enabled
```

4.3.14 Task: Use a Browser to Access Web Page

To test the Web Server, point your browser at the IP address defined in the VirtualIP resource above:

Figure 9: Web Browser Connection to Cluster Resource

4.3.15 Task: Test Failover

On rh7cn1 node place the node into standby mode:

```
# pcs cluster standby rh7cn1.sinenomine.net
```

Use the status command to verify that the apachegroup resources are now running on the other node:

```
# pcs status
Cluster name: rh7cluster
Last updated: Fri Oct  3 16:04:51 2014
Last change: Fri Oct  3 15:54:53 2014 via crm_attribute on
rh7cn1.sinenomine.net
Stack: corosync
Current DC: rh7cn1.sinenomine.net (1) - partition with quorum
Version: 1.1.10-29.el7-368c726
2 Nodes configured
11 Resources configured


Online: [ rh7cn1.sinenomine.net rh7cn2.sinenomine.net ]

Full list of resources:

:
:
 Resource Group: apachegroup
     VirtualIP (ocf::heartbeat:IPaddr2):  Started rh7cn2.sinenomine.net
     Website   (ocf::heartbeat:apache):   Started rh7cn2.sinenomine.net
     httplvm   (ocf::heartbeat:LVM):   Started rh7cn2.sinenomine.net
     http_fs   (ocf::heartbeat:Filesystem):  Started
rh7cn2.sinenomine.net
```


:
:

Refresh the browser to verify that the Web Server is still available.

5 glusterfs Configuration

GlusterFS is a clustered file system, capable of scaling to several petabytes. It aggregates various storage bricks over Infiniband RDMA or TCP/IP and interconnects into one large parallel network file system. Storage bricks can be made of any commodity hardware, such as x86-64 server with SATA-II RAID and Infiniband HBA.

GlusterFS is fully POSIX compliant file system. On the client side, it has dependency on FUSE package; on the server side, it works seamlessly on different operating systems. Currently it is supported on GNU/Linux and Solaris.

The following tasks enable you to set up glusterfs on the two nodes created earlier.

5.1 Task: Install ntp

On both nodes:

```
# yum install ntp glusterfs-server glusterfs-fuse glusterfs nfs-utils
```

5.2 Task: Enable ntp service

On both nodes:

```
# systemctl enable ntpd
# systemctl start ntpd
```

5.3 Task: Check ntp status

On both nodes:

```
# timedatectl status
      Local time: Mon 2014-10-06 14:15:31 EDT
  Universal time: Mon 2014-10-06 18:15:31 UTC
        Timezone: America/New_York (EDT, -0400)
     NTP enabled: yes
NTP synchronized: yes
 RTC in local TZ: no
      DST active: yes
 Last DST change: DST began at
                  Sun 2014-03-09 01:59:59 EST
                  Sun 2014-03-09 03:00:00 EDT
 Next DST change: DST ends (the clock jumps one hour backwards) at
                  Sun 2014-11-02 01:59:59 EDT
                  Sun 2014-11-02 01:00:00 EST
```

If NTP enabled is set to “no” enter:

```
# timedatectl set-ntp 1
```

5.4 Task: Create firewall rules for glusterfs

On both nodes, create the file `/etc/firewalld/services/gluster.xml` containing:

Figure 10: gluster Firewall Service File

```
<?xml version="1.0" encoding="utf-8"?>
<service>
  <short>glusterfs</short>
  <description>GlusterFS is a clustered file-system capable of scaling to
several petabytes. It aggregates various storage bricks over Infiniband RDMA
or TCP/IP interconnect into one large parallel network file system. GlusterFS
is one of the most sophisticated file systems in terms of features and
extensibility.  It borrows a powerful concept called Translators from GNU Hurd
kernel. Much of the code in GlusterFS is in user space and easily manageable.
  </description>
  <port port="24007" protocol="tcp" />
  <port port="24009" protocol="tcp" />
  <port port="24008" protocol="tcp" />
  <port port="24010" protocol="tcp" />
  <port port="49152" protocol="tcp" />
  <port port="49153" protocol="tcp" />
  <port port="49154" protocol="tcp" />
  <port port="49155" protocol="tcp" />
</service>
```

5.5 Task: Enable firewall rules

On both nodes:

```
# firewall-cmd --permanent --add-service=gluster
# systemctl restart firewalld
```

5.6 Task: Install SELinux Rules for glusterd

If you haven't installed an SElinux policy such as selinux-policy-targeted, then on both nodes:

```
# yum install selinux-policy-targeted -y
```

5.7 Task: Create rsyslog configuration

On both nodes:

```
# mv /etc/rsyslog.d/gluster.conf.example /etc/rsyslog.d/gluster.conf
# systemctl restart rsyslog
```

5.8 Task: Create mount point for gluster

On both nodes:

```
# mkdir /mnt/gluster
```

5.9 Task: Probe for Peer

On rh7cn1:

```
# gluster peer probe rh7cn2
```

5.10 Task: Activate gluster Resource Disk on RH7CN1 and RH7CN2

If the gluster disk was not added at RHEL installation time, then make it available now on both nodes:

```
# cio_ignore –R
# chccwdev –e 0.0.0300
```

5.11 Task: Format gluster volume

On both nodes:

```
# dasdfmt –b 4096 –y /dev/dasdf
# fdasd -a /dev/dasdf
```

5.12 Task: Create Physical Volume

On both nodes:

```
# pvcreate /dev/dasdf1
```

5.13 Task: Create a non-clustered Logical Volume

On both nodes:

```
# vgcreate –cn vg_gluster /dev/dasdf1
```

5.14 Task: Activate the New Volume Group

On both nodes:

```
# vgchange –ay vg_gluster
```

5.15 Task: Create gluster Logical Volume

On both nodes:

```
# lvcreate -L120M -n gluster_vol vg_gluster
```

5.16 Task: Make an XFS File System for gluster

On both nodes:

```
# mkfs.xfs /dev/mapper/vg_gluster-gluster_vol
```

5.17 Task: Add gluster Volume to /etc/fstab

On both nodes, add the following line to `/etc/fstab`:

```
/dev/mapper/vg_gluster-gluster_vol /mnt/gluster/   xfs   defaults   1 3
```

5.18 Task: Mount Volume

On both nodes:

```
# mount -a
```

5.19 Task: Create a Directory for a Brick

On both nodes:

```
# mkdir /mnt/gluster/brick
```

5.20 Task: Create a gluster Volume

On one of the nodes:

```
# gluster volume create vol1 replica 2 rh7cn1:/mnt/gluster/brick
rh7cn2:/mnt/gluster/brick
```

Command explanation: Define a volume that is replicated over two nodes using an area on the xfs volume dedicated to gluster.

5.21 Task: Start the gluster Volume

On one of the nodes:

```
# gluster volume start vol1
```

5.22 Task: Create a glusterfs Mount Point

On both of the nodes:

```
# mkdir /mnt/store
```

5.23 Task: Mount a glusterfs Volume

On both nodes:

```
# mount -t glusterfs localhost:/vol1 /mnt/store
```

5.24 Task: Verify File System is Mounted

On both nodes:

```
# df –hT
Filesystem                          Type            Size  Used Avail Use% Mounted on
/dev/mapper/rhel_rh7cn1-root        xfs             3.7G  1.2G  2.5G  32% /
devtmpfs                            devtmpfs        366M     0  366M   0% /dev
tmpfs                               tmpfs           371M   76M  296M  21% /dev/shm
tmpfs                               tmpfs           371M   15M  357M   4% /run
tmpfs                               tmpfs           371M     0  371M   0% /sys/fs/cgroup
/dev/mapper/rhel_rh7cn1-boot        xfs             494M  104M  391M  21% /boot
/dev/mapper/vg_cluster-ha_lv        gfs2            300M   35M  266M  12% /mnt/gfs2-demo
/dev/mapper/vg_gluster-gluster_vol xfs              114M  6.3M  108M   6% /mnt/gluster
localhost:/vol1                     fuse.glusterfs  114M  6.3M  108M   6% /mnt/store
```

5.25 Task: Create a File in glusterfs

On one node:

```
# echo “ABCDEF” >/mnt/store/test.file
```

5.26 Task: Verify File Exists on Other Node

On both nodes:

```
# ls –l /mnt/store
total 1
-rw-r--r--. 1 root root 7 Oct  7 15:26 test.file

# cat /mnt/store/test.file
ABCDEF
```

5.27 Task: Shutdown and Disable gluster

Now that we’ve verified that gluster is working, we want to make gluster a cluster resource that pacemaker will manage. On both of the nodes:

```
# systemctl stop glusterfsd
# systemctl stop glusterd
# systemctl disable glusterd
```

5.28 Task: Define the gluster Service to Pacemaker

On one of the nodes create a copy of the live configuration into a file `gluster_cfg` to which we will make changes:

```
# pcs cluster cib gluster_cfg
```

5.28.1 Task: Define the bricks as Filesystem Resources

On the same node:

```
# pcs -f gluster_cfg resource create brick1 Filesystem
device=/dev/mapper/vg_gluster-gluster_vol directory=/mnt/gluster
fstype=xfs
# pcs -f gluster_cfg resource create brick2 Filesystem
device=/dev/mapper/vg_gluster-gluster_vol directory=/mnt/gluster
fstype=xfs
# pcs -f gluster_cfg constraint location brick1 prefers
rh7cn1.sinenomine.net
# pcs -f gluster_cfg constraint location brick2 prefers
rh7cn2.sinenomine.net
# pcs -f gluster_cfg constraint colocation add brick1 with brick2 –
INFINITY
```

Command explanation:

1. Define brick1 to represent the volume we will use as the backing storage for the gluster volume on rh7cn1: the xfs file system on device `/dev/mapper/vg_gluster-gluster_vol` will be mounted at `/mnt/gluster`
2. Likewise, define brick2 for use on rh7cn2: the xfs file system on device `/dev/mapper/vg_gluster-gluster_vol` will be mounted at `/mnt/gluster`
3. Set a constraint so that brick1 will only start on rh7cn1
4. Same for brick2 on rh7cn2
5. Ensure that the two bricks never start on the same system

5.28.2 Task: Create the gluster daemon Resource

On the same node:

```
# pcs -f gluster_cfg resource create glusterd ocf:glusterfs:glusterd op
monitor interval=20s start-delay=15s --clone meta interleave=true
ordered=true --force op timeout=120 op stop timeout=120
```

Command explanation: Define the daemon as a clone set so that it will be started on both nodes of the cluster

```
# pcs -f gluster_cfg resource create gvol1 ocf:glusterfs:volume
volname=vol1
# pcs -f gluster_cfg resource create gvol2 ocf:glusterfs:volume
volname=vol1
# pcs -f gluster_cfg constraint colocation add gvol1 with brick1
```

```
# pcs -f gluster_cfg constraint colocation add gvol2 with brick2
# pcs -f gluster_cfg constraint order glusterd-clone then gvol1
# pcs -f gluster_cfg constraint order glusterd-clone then gvol2
```

Command explanation:

1. Create the gluster volume resource gvol1 that we will associate with brick1 that represents the vol1 gluster volume created earlier
2. Create the gluster volume resource gvol2 that we will associate with brick2 that represents the vol1 gluster volume created earlier
3. Constrain gvol1 to co-locate with brick1
4. Constrain gvol2 to co-locate with brick2
5. Start the gvol1 volume after the gluster daemon
6. Start the gvol2 volume after the gluster daemon

5.28.3 Task: Create gluster Filesystem Resources

On the same node:

```
# pcs -f gluster_cfg resource create store1 Filesystem
device=localhost:/vol1 directory=/mnt/store fstype=glusterfs
# pcs -f gluster_cfg resource create store2 Filesystem
device=localhost:/vol1 directory=/mnt/store fstype=glusterfs
# pcs -f gluster_cfg constraint colocation add store1 with brick1
# pcs -f gluster_cfg constraint colocation add store2 with brick2
# pcs -f gluster_cfg constraint order gvol1 then store1
# pcs -f gluster_cfg constraint order gvol2 then store2
```

Command explanation:

1. Define resource store1 to represent the gluster file system mounted as `localhost:/vol1` on `/mnt/store`
2. Define resource store2 to represent the gluster file system mounted as `localhost:/vol1` on `/mnt/store`
3. Constrain store1 to co-locate with brick1
4. Constrain store2 to co-locate with brick2
5. Start the store1 resource after gvol1 has started
6. Start the store2 resource after gvol2 has started

5.28.4 Task: Make the new cluster configuration live

On the same node:

```
# pcs cluster cib-push gluster_cfg
```

Command explanation: This command will push the definitions created in the previous sections that are in the file `gluster_cfg` to the live configuration. Pacemaker will then start the resources.

6 drbd Installation

Distributed Replicating Block Device (DRBD), is a technology that takes raw storage from two nodes and keeps their data synchronized in real time. It is sometimes described as “network RAID Level 1”, and that is conceptually accurate. In this tutorial's cluster, DRBD will be used to provide that back-end storage as a cost-effective alternative to a traditional SAN device.

DRBD is, fundamentally, a raw block device. If you have ever used `mdadm` to create a software RAID array, then you will be familiar with this.

Think of it this way:

With traditional software raid, you would take:

- `/dev/sda5 + /dev/sdb5 -> /dev/md0`

With DRBD, you have this:

- `node1:/dev/sda5 + node2:/dev/sda5 -> both:/dev/drbd0`

In both cases, as soon as you create the new `md0` or `drbd0` device, you pretend like the member devices no longer exist. You format a filesystem onto `/dev/md0`, use `/dev/drbd0` as an LVM physical volume, and so on.

The main difference with DRBD is that the `/dev/drbd0` will always be the same on both nodes. If you write something to node 1, it is instantly available on node 2, and vice versa. Of course, this means that what ever you put on top of DRBD has to be “cluster aware”. That is to say, the program or file system using the new `/dev/drbd0` device has to understand that the contents of the disk might change because of another node. [9]

6.1 Task: Install drbd Packages

On both nodes:

```
# yum install drbd-kmod drbd-utils drbd-pacemaker -y
```

6.2 Task: Install SELinux Policies

If you haven’t performed this step already (see “5.6 Task: Install SELinux Rules for glusterd” on page 5-11, then on both nodes:

```
# yum install selinux-policy-targeted –y
# semodule -i /etc/selinux/targeted/modules/active/modules/drbd.pp
# semanage permissive -a drbd_t
```

Command explanation: The `semodule` command will ensure the drbd policy is active in the system. The `semanage` command is used to configure certain elements of SELinux policy without requiring modification to or recompilation from policy sources.

6.3 Task: Create drbd Configuration

On both nodes, create `/etc/drbd.d/disk1.res` with the following contents:

```
resource disk1
{
        startup {
                wfc-timeout 30;
                outdated-wfc-timeout 20;
                degr-wfc-timeout 30;
        }

        net {
                cram-hmac-alg sha1;
                shared-secret sync_disk;
        }

        syncer {
                rate 100M;
                verify-alg sha1;
        }

        on rh7cn1.sinenomine.net {
                device minor 1;
                disk /dev/disk/by-path/ccw-0.0.0301-part1;
                address 172.17.16.147:7789;
                meta-disk internal;
        }

        on rh7cn2.sinenomine.net {
                device minor 1;
                disk /dev/disk/by-path/ccw-0.0.0301-part1;
                address 172.17.16.148:7789;
                meta-disk internal;
        }
}
```

6.4 Task: Create Firewall Rules for drbd

On both nodes, create the file `/etc/firewalld/services/drbd.xml` containing the following:

```
<?xml version="1.0" encoding="utf-8"?>
<service>
  <short>Distributed Replicated Block Device</short>
```

```
  <description>This allows you to use the Distributed Replicated Block
Device</description>
  <port protocol="tcp" port="7789"/>
</service>
```

On both nodes:

```
# firewall-cmd --permanent --add-service drbd
```

6.5 Task: Activate drbd Resource Disk on RH7CN1 and RH7CN2

If the drbd disk was not added at RHEL installation time, then make it available now on both nodes:

```
# cio_ignore –R
# chccwdev –e 0.0.0301
```

6.6 Task: Create the drbd Metadata

On both nodes:

```
# drbdadm create-md disk1
```

You should see output similar to this and a prompt to answer “yes”:

```
md_offset 147439616
al_offset 147406848
bm_offset 147398656

Found xfs filesystem
      143944 kB data area apparently used
      143944 kB left usable by current configuration

Even though it looks like this would place the new meta data into
unused space, you still need to confirm, as this is only a guess.

Do you want to proceed?
[need to type 'yes' to confirm] yes

initializing activity log
NOT initializing bitmap
Writing meta data...
New drbd meta data block successfully created.
```

6.7 Task: Set the Primary Node

On one of the nodes you wish to make primary:

```
# /sbin/drbdadm -- --overwrite-data-of-peer primary disk1
```

6.8 Task: Start the drbd Service

On both nodes:

```
# systemctl enable drbd
# systemctl start drbd
```

6.9 Task: Wait to Syncrhonize

On both nodes:

```
# cat /proc/drbd
```

On the primary node you should see something similar to this:

```
version: 8.4.3 (api:1/proto:86-101)
srcversion: FC2645CB3DC006D14926752

 1: cs:SyncSource ro:Primary/Secondary ds:UpToDate/Inconsistent C r-----
    ns:142924 nr:0 dw:0 dr:144672 al:0 bm:8 lo:0 pe:1 ua:4 ap:0 ep:1 wo:f oos:1608
    [===================>] sync'ed:100.0% (1608/143944)K
    finish: 0:00:00 speed: 5,588 (5,080) K/sec
```

On the secondary node:

```
version: 8.4.3 (api:1/proto:86-101)
srcversion: FC2645CB3DC006D14926752

 1: cs:Connected ro:Secondary/Primary ds:UpToDate/UpToDate C r-----
    ns:0 nr:143944 dw:143944 dr:0 al:0 bm:9 lo:0 pe:0 ua:0 ap:0 ep:1 wo:f oos:0
```

The key is to see the 100% on the primary node and UpToDate on the secondary.

6.10 Task: Create a File System on the drbd Volume

On the primary node:

```
# mkfs.xfs /dev/drbd1
```

6.11 Task: Mount the File System

On the primary node:

```
# mount /dev/drbd1 /mnt/drbd
```

6.12 Task: Write to a Test File

On the primary node:

```
# tail -20 /var/log/messages >/mnt/drbd/test.file
# cat /mnt/drbd/test.file
```

```
Oct 23 13:44:42 rh7cn1 kernel: block drbd1: peer
0000000000000004:0000000000000000:0000000000000000:0000000000000000 bits:35986 flags:0
Oct 23 13:44:42 rh7cn1 kernel: block drbd1: uuid_compare()=2 by rule 30
Oct 23 13:44:42 rh7cn1 kernel: block drbd1: Becoming sync source due to disk states.
Oct 23 13:44:42 rh7cn1 kernel: block drbd1: Writing the whole bitmap, full sync
required after drbd_sync_handshake.
Oct 23 13:44:42 rh7cn1 kernel: block drbd1: bitmap WRITE of 2 pages took 0 jiffies
Oct 23 13:44:42 rh7cn1 kernel: block drbd1: 141 MB (35986 bits) marked out-of-sync by
on disk bit-map.
Oct 23 13:44:42 rh7cn1 kernel: block drbd1: conn( Connected -> WFBitMapS )
Oct 23 13:44:42 rh7cn1 kernel: block drbd1: send bitmap stats [Bytes(packets)]: plain
0(0), RLE 21(1), total 21; compression: 99.6%
Oct 23 13:44:42 rh7cn1 kernel: block drbd1: receive bitmap stats [Bytes(packets)]:
plain 0(0), RLE 21(1), total 21; compression: 99.6%
Oct 23 13:44:42 rh7cn1 kernel: block drbd1: helper command: /sbin/drbdadm before-
resync-source minor-1
Oct 23 13:44:42 rh7cn1 kernel: block drbd1: helper command: /sbin/drbdadm before-
resync-source minor-1 exit code 0 (0x0)
Oct 23 13:44:42 rh7cn1 kernel: block drbd1: conn( WFBitMapS -> SyncSource )
Oct 23 13:44:42 rh7cn1 kernel: block drbd1: Began resync as SyncSource (will sync
143944 KB [35986 bits set]).
Oct 23 13:44:42 rh7cn1 kernel: block drbd1: updated sync UUID
1EE7DD96ABF615D3:4E0446B8BCC3E101:0000000000000004:0000000000000000
Oct 23 13:45:11 rh7cn1 kernel: block drbd1: Resync done (total 28 sec; paused 0 sec;
5140 K/sec)
Oct 23 13:45:11 rh7cn1 kernel: block drbd1: updated UUIDs
1EE7DD96ABF615D3:0000000000000000:4E0446B8BCC3E101:0000000000000004
Oct 23 13:45:11 rh7cn1 kernel: block drbd1: conn( SyncSource -> Connected ) pdsk(
Inconsistent -> UpToDate )
Oct 23 13:51:24 rh7cn1 systemd-udevd: inotify_add_watch(7, /dev/dm-4, 10) failed: No
such file or directory
Oct 23 13:54:43 rh7cn1 kernel: XFS (dm-4): Mounting Filesystem
Oct 23 13:54:44 rh7cn1 kernel: XFS (dm-4): Ending clean mount
```

6.13 Task: Check the Device on Secondary Node

On the primary node, unmount the device and make it a secondary:

```
# umount /mnt/drbd
# drbdadm secondary disk1
```

On the secondary node, make the device primary and mount it:

```
# drbdadm primary disk1
# mount /dev/drbd1 /mnt/drbd
```

On the secondary, verify the file created on the primary is present:

```
# ls –l /mnt/drbd
total 4
-rw-r--r--. 1 root root 2076 Oct 23 13:55 test.file
```

6.14 Making drbd as a Cluster Resource

The following steps will make the drbd disk a cluster resource so that a failure of the primary node will cause failover to the secondary.

6.14.1 Task: Ensure drbd is not started at boot time

On both nodes:

```
# systemctl stop drbd
# systemctl disable drbd
```

6.14.2 Task: Add Resources to Cluster

On one of the nodes:

```
# pcs cluster cib drbd_cfg
# pcs –f drbd_cfg resource create GDPS ocf:linbit:drbd
drbd_resource=disk1 op monitor interval=60s
# pcs –f drbd_cfg resource master GDPSClone GDPS master-max=1 master-
node-max=1 clone-max=2 clone-node-max=1 notify=true
# pcs –f drbd_cfg resource create GDPSFS Filesystem device="/dev/drbd1"
directory="/mnt/drbd" fstype="xfs"
# pcs –f drbd_cfg constraint colocation add GDPSFS GDPSClone INFINITY
with-rsc-role=Master
# pcs –f drbd_cfg constraint order promote GDPSClone then start GDPSFS
# pcs cluster cib-push drbd_cfg
# rm drbd_cfg
```

Command explanations: These commands:

1. Grab a copy of the active configuration so we can make changes discretely without affecting the live system
2. Define a DRDB resource call GDPS that uses the `disk1` resource created earlier (see “6.3 Task: Create drbd Configuration” on page 6-2)
3. Define the resource as multi-state, one node will be the master and there will be a maximum of 2 clones of this resource, with a maximum of one clone being active on any node
4. Create an xfs file system residing on the `/dev/drbd1` device node that will be mounted at `/mnt/drbd` on the master node
5. Constrain the file system to reside on the same node as the GDPS resource
6. Start the GDPS resource before starting the file system resource
7. Push the local changes to the live configurations

7 Now that Configuration is Complete

If you have followed all the steps outlined in this document your two-node cluster should be up and running with multiple resources. The pcs status command should return something looking like:

Figure 11: Cluster Status with all resources defined and started

```
Cluster name:
Last updated: Tue Oct 28 17:14:18 2014
Last change: Tue Oct 28 16:45:27 2014 via cibadmin on
rh7cn1.sinenomine.net
Stack: corosync
Current DC: rh7cn1.sinenomine.net (1) - partition with quorum
Version: 1.1.10-29.el7-368c726
2 Nodes configured
22 Resources configured


Online: [ rh7cn1.sinenomine.net rh7cn2.sinenomine.net ]

Full list of resources:

 ZVMPOWER  (stonith:fence_zvm): Started rh7cn1.sinenomine.net
 Clone Set: dlm-clone [dlm]
     Started: [ rh7cn1.sinenomine.net rh7cn2.sinenomine.net ]
 Resource Group: apachegroup
     VirtualIP (ocf::heartbeat:IPaddr2):  Started rh7cn2.sinenomine.net
     Website  (ocf::heartbeat:apache):  Started rh7cn2.sinenomine.net
     httplvm  (ocf::heartbeat:LVM):  Started rh7cn2.sinenomine.net
     http_fs  (ocf::heartbeat:Filesystem):  Started
rh7cn2.sinenomine.net
 Clone Set: clvmd-clone [clvmd]
     Started: [ rh7cn1.sinenomine.net rh7cn2.sinenomine.net ]
 Clone Set: clusterfs-clone [clusterfs]
     Started: [ rh7cn1.sinenomine.net rh7cn2.sinenomine.net ]
 Master/Slave Set: GDPSClone [GDPS]
     Masters: [ rh7cn1.sinenomine.net ]
     Slaves: [ rh7cn2.sinenomine.net ]
 GDPSFS  (ocf::heartbeat:Filesystem):  Started rh7cn1.sinenomine.net
 brick1  (ocf::heartbeat:Filesystem):  Started rh7cn1.sinenomine.net
 brick2  (ocf::heartbeat:Filesystem):  Started rh7cn2.sinenomine.net
 Clone Set: glusterd-clone [glusterd]
     Started: [ rh7cn1.sinenomine.net rh7cn2.sinenomine.net ]
 gvol1  (ocf::glusterfs:volume):  Started rh7cn1.sinenomine.net
 gvol2  (ocf::glusterfs:volume):  Started rh7cn2.sinenomine.net
 store1  (ocf::heartbeat:Filesystem):  Started rh7cn1.sinenomine.net
 store2  (ocf::heartbeat:Filesystem):  Started rh7cn2.sinenomine.net
```

```
PCSD Status:
  rh7cn1.sinenomine.net: Online
  rh7cn2.sinenomine.net: Online

Daemon Status:
  corosync: active/enabled
  pacemaker: active/enabled
  pcsd: active/enabled
```

The output of your `df -hT` command on the DRBD primary node should look something like this:

Figure 12: File System Configuration Display

```
Filesystem                          Type            Size  Used Avail Use% Mounted on
/dev/mapper/rhel_rh7cn1-root        xfs             3.7G  1.7G  2.0G  46% /
devtmpfs                            devtmpfs        366M     0  366M   0% /dev
tmpfs                               tmpfs           371M   76M  296M  21% /dev/shm
tmpfs                               tmpfs           371M   38M  333M  11% /run
tmpfs                               tmpfs           371M     0  371M   0% /sys/fs/cgroup
/dev/mapper/rhel_rh7cn1-boot        xfs             494M  104M  391M  21% /boot
/dev/mapper/vg_cluster-ha_lv        gfs2            300M   35M  266M  12% /mnt/gfs2-demo
/dev/drbd1                          xfs             135M  7.3M  128M   6% /mnt/drbd
/dev/mapper/vg_gluster-gluster_vol xfs              114M  6.3M  108M   6% /mnt/gluster
localhost:/vol1                     fuse.glusterfs  114M  6.3M  108M   6% /mnt/store
```

7.1 Configuration Refinement

The network configuration used in this sample is very simple. In a production environment the network should be configured functionally. That is, user traffic should use a separate subnet to the cluster infrastructure, and to the network-based file system such as glusterfs or drbd.

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 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 4: fence_zvm Options

Option	Description
Command line Options	
-o --action action	Fencing action: "onoff" or “reboot” - fence off device; "metadata" - display device metadata; “status” – Return status of a virtual machine
-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
Standard Input Options	
agent = agent	This option is used by fence_node(8) and is ignored by fence_zvm.
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

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 XXXXXXX 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 a power Fencing agent used on a GFS virtual machine in a 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 5: fence_zvmip Options

Option	Description
Command line Options	
-o --action action	Fencing action: "onoff" or “reboot” - fence off device; "metadata" - display device metadata; “status” – Return status of a virtual machine
-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 XXXXXXX 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 RH7CN1, RH7CN2 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:

```
/* 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, which also contains definitions used in the RHEL 6 HA offering but will do no harm in a RHEL 7 environment.

```
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).

```
/* 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

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 pcs Command Reference

The following section is taken from the pcs man page.

10.1 Name

pcs - pacemaker/corosync configuration system

10.2 Synopsis

```
pcs [-f file] [-h] [commands]...
```

10.3 Description

Control and configure pacemaker and corosync.

10.4 Options

Table 6: pcs Command Options

Option	Description
`-h, --help`	Display usage and exit
`-f file`	Perform actions on file instead of active CIB
`--debug`	Print all network traffic and external commands run
`--version`	Print pcs version information

10.5 Subcommands

Table 7: pcs Subcommands

Command	Description
`resource`	Manage cluster resources
`cluster`	Configure cluster options and nodes
`stonith`	Configure fence devices
`property`	Set pacemaker properties
`constraint`	Set resource constraints

`status`	View cluster status
`config`	Print full cluster configuration

10.5.1 resource Subcommand

The resource subcommand is used to manage cluster resources.

10.5.1.1 show

```
show [resource id] [--full] [--groups]
```

Show all currently configured resources or if a resource is specified show the options for the configured resource. If `--full` is specified all configured resource options will be displayed. If `--groups` is specified, only show groups (and their resources).

10.5.1.2 list

```
list [<standard|provider|type>] [--nodesc]
```

Show list of all available resources, optionally filtered by specified type, standard or provider. If `--nodesc` is used then descriptions of resources are not printed.

10.5.1.3 describe

```
describe <standard:provider:type|type>
```

Show options for the specified resource

10.5.1.4 create

```
create <resource id> <standard:provider:type|type> [resource options]
[op <operation action> <operation options> [<operation action>
<operation options>]...] [meta <meta options>...] [--clone <clone
options> | --master <master options> | --group <group name>]
```

Create specified resource. If `--clone` is used a clone resource is created if `--master` is specified a master/slave resource is created. If `--group` is specified the resource is added to the group named.

Example: Create a new resource called 'VirtualIP' with IP address 192.168.0.99, netmask of 32, monitored everything 30 seconds, on eth2:

```
pcs resource create VirtualIP ocf:heartbeat:IPaddr ip=192.168.0.99
cidr_netmask=32 nic=eth2 op monitor interval=30s
```

10.5.1.5 delete

```
delete <resource id|group id|master id|clone id>
```

Deletes the resource, group, master or clone (and all resources within the group/master/clone).

10.5.1.6 enable

```
enable <resource id> [--wait[=n]]
```

Allow the cluster to start the resource. Depending on the rest of the configuration (constraints, options, failures, etc.), the resource may remain stopped. If `--wait` is specified, pcs will wait up to 30 seconds (or '*n*' seconds) for the resource to start and then return 0 if the resource is started, or 1 if the resource has not yet started.

10.5.1.7 disable

```
disable <resource id> [--wait[=n]]
```

Attempt to stop the resource if it is running and forbid the cluster from starting it again. Depending on the rest of the configuration (constraints, options, failures, etc.), the resource may remain started. If `--wait` is specified, pcs will wait up to 30 seconds (or '*n*' seconds) for the resource to stop and then return 0 if the resource is stopped or 1 if the resource has not stopped.

10.5.1.8 debug

```
debug-start <resource id> [--full]
```

This command will force the specified resource to start on this node ignoring the cluster recommendations and print the output from starting the resource. Using `--full` will give more detailed output. This is mainly used for debugging resources that fail to start.

10.5.1.9 move

```
move <resource id> [destination node] [--master]
```

Move resource off current node (and optionally onto destination node). If `--master` is used the scope of the command is limited to the master role and you must use the master id (instead of the resource id).

10.5.1.10 ban

```
ban <resource id> [node] [--master]
```

Prevent the resource id specified from running on the node (or on the current node it is running on if no node is specified). If `--master` is used the scope of the command is limited to the master role and you must use the master id (instead of the resource id).

10.5.1.11 clear

```
clear <resource id> [node] [--master]
```

Remove constraints created by move and/or ban on the specified resource (and node if specified). If `--master` is used the scope of the command is limited to the master role and you must use the master id (instead of the resource id).

10.5.1.12 standards

```
standards
```

List available resource agent standards supported by this installation. (OCF, LSB, etc.)

10.5.1.13 providers

```
providers
```

List available OCF resource agent providers.

10.5.1.14 agents

```
agents [standard[:provider]]
```

List available agents optionally filtered by standard and provider

10.5.1.15 update

```
update <resource id> [resource options] [op [<operation action> <operation options>]...] [meta <meta operations>...]
```

Use the Add/Change options to specified resource, clone or multi-state resource. If an operation (op) is specified it will update the first found operation with the same action on the specified resource, if no operation with that action exists then a new operation will be created (**WARNING**: all current options on the update op will be reset if not specified). If you want to create multiple monitor operations you should use the add_operation & remove_operation commands.

10.5.1.16 op add

```
op add <resource id> <operation action> [operation properties]
```

Add operation for specified resource.

10.5.1.17 op remove

```
op remove <resource id> <operation action> <operation properties>
```

Remove specified operation (note: you must specify the exact operation properties to properly remove an existing operation).

```
op remove <operation id>
```

Remove the specified operation id.

10.5.1.18 op defaults

```
op defaults [options]
```

Set default values for operations, if no options are passed, lists currently configured defaults

10.5.1.19 meta

```
meta <resource id | group id | master id | clone id> <meta options>
```

Add specified options to the specified resource, group, master/slave or clone. Meta options should be in the format of name=value, options may be removed by setting an option without a value.

Example: `pcs resource meta TestResource failure-timeout=50 stickiness=`

10.5.1.20 group add

```
group add <group name> <resource id> [resource id] ... [resource id]
```

Add the specified resource to the group, creating the group if it does not exist. If the resource is present in another group it is moved to the new group.

10.5.1.21 group remove

```
group remove <group name> <resource id> [resource id] ... [resource id]
```

Remove the specified resource(s) from the group, removing the group if it no resources remain.

10.5.1.22 ungroup

```
ungroup <group name> [resource id] ... [resource id]
```

Remove the group (note: this does not remove any resources from the cluster) or if resources are specified, remove the specified resources from the group.

10.5.1.23 clone

```
clone <resource id | group id> [clone options]...
```

Setup up the specified resource or group as a clone.

10.5.1.24 unclone

```
unclone <resource id | group name>
```

Remove the clone that contains the specified group or resource (the resource or group will not be removed)

10.5.1.25 master

```
master [<master/slave name>] <resource id | group name> [options]
```

Configure a resource or group as a multi-state (master/slave) resource. Note: to remove a master you must remove the resource/group it contains.

```
manage <resource id> ... [resource n]
```

Set resources listed to “managed” mode (default).

10.5.1.26 unmanage

```
unmanage <resource id> ... [resource n]
```

Set resources listed to unmanaged mode.

10.5.1.27 defaults

```
defaults [options]
```

Set default values for resources, if no options are passed, lists currently configured defaults.

10.5.1.28 cleanup

```
cleanup <resource id>
```

Cleans up the resource in the `lrmd` (useful to reset the resource status and failcount). This tells the cluster to forget the operation history of a resource and re-detect its current state. This can be useful to purge knowledge of past failures that have since been resolved.

10.5.1.29 failcount

```
failcount show <resource id> [node]
```

Show current failcount for specified resource from all nodes or only on specified node

```
failcount reset <resource id> [node]
```

Reset failcount for specified resource on all nodes or only on specified node. This tells the cluster to forget how many times a resource has failed in the past. This may allow the resource to be started or moved to a more preferred location.

10.5.2 cluster Subcommand

Configure cluster options and nodes.

10.5.2.1 auth

```
auth [node] [...] [-u username] [-p password] [--local] [--force]
```

Authenticate pcs to `pcsd` on nodes specified, or on all nodes configured in `corosync.conf` if no nodes are specified (authorization tokens are stored in `~/.pcs/tokens` or `/var/lib/pcsd/tokens` for root). By default all nodes are also authenticated to each other, using `--local` only authenticates the local node (and does not authenticate the remote nodes with each other).

Using `--force` forces re-authentication to occur.

10.5.2.2 setup

```
setup [--start] [--local] [--enable] –name <cluster name>
<node1[,node1-altaddr]> [node2[,node2-altaddr]] [..] [--transport
<udpu|udp>] [--rrpmode active|passive] [--addr0  <addr/net> [[[--mcast0
<address>]  [--mcastport0  <port>] [--ttl0 <ttl>]] | [--broadcast0]] [-
-addr1 <addr/net> [[[--mcast1    <address>]    [--mcastport1    <port>]
[--ttl1    <ttl>]] | [--broadcast1]]]] [--wait_for_all=<0|1>] [--
auto_tie_breaker=<0|1>]   [--last_man_standing=<0|1> [--
last_man_standing_window=<time in ms>]] [--token   <timeout>] [--join
<timeout>] [--consensus <timeout>] [--miss_count_const <count>] [--
fail_recv_const <failures>]
```

Configure corosync and synchronize configuration out to listed nodes.

`--local` will only perform changes on the local node,

`--start` will also start the cluster on the specified nodes,

`--enable` will enable corosync and pacemaker on node startup,

`--transport` allows specification of corosync transport (default: udpu).

The `--wait_for_all`, `--auto_tie_breaker`, `--last_man_standing`, `--last_man_standing_window` options are all documented in corosync's vote-quorum(5) man page.

`--ipv6` will configure corosync to use ipv6 (instead of ipv4)

`--token <timeout>` sets time in milliseconds until a token loss is declared after not receiving a token (default 1000 ms)

`--join <timeout>` sets time in milliseconds to wait for join messages (default 50 ms)

`--consensus <timeout>` sets time in milliseconds to wait for consensus to be achieved before starting a new round of membership configuration (default 1200 ms)

`--miss_count_const <count>` sets the maximum number of times on receipt of a token a message is checked for retransmission before a retransmission occurs (default 5 messages)

--fail_recv_const <failures> specifies how many rotations of the token without receiving any messages when messages should be received may occur before a new configuration is formed (default 2500 failures)

Configuring Redundant Ring Protocol (RRP)

When using udpu (the default) specifying nodes, specify the ring 0 address first followed by a ',' and then the ring 1 address.

Example: `pcs cluster setup --name cname nodeA-0,nodeA-1 nodeB-0,nodeB-1`

When using udp, using `--addr0` and `--addr1` will allow you to configure rrp mode for corosync. It's recommended to use a network (instead of IP address) for `--addr0` and `--addr1` so the same corosync.conf file can be used around thecluster. `--mcast0` defaults to 239.255.1.1 and `--mcast1` defaults to 239.255.2.1, `--mcastport0/1` default to 5405 and ttl defaults to 1. If `--broadcast` is specified, `--mcast0/1`, --mcastport0/1 & `--ttl0/1` are ignored.

10.5.2.3 start

```
start [--all] [node] [...]
```

Start corosync & pacemaker on specified node(s), if a node is not specified then corosync & pacemaker are started on the local node. If --all is specified then corosync & pacemaker are started on all nodes.

10.5.2.4 stop

```
stop [--all] [node] [...]
```

Stop corosync & pacemaker on specified node(s), if a node is not specified then corosync & pacemaker are stopped on the local node. If --all is specified then corosync & pacemaker are stopped on all nodes.

10.5.2.5 kill

`kill`

Force corosync and pacemaker daemons to stop on the local node (performs kill -9).

10.5.2.6 enable

`enable [--all] [node] [...]`

Configure corosync & pacemaker to run on node boot on specified node(s), if node is not specified then corosync & pacemaker are enabled on the local node. If `--all` is specified then corosync & pacemaker are enabled on all nodes.

10.5.2.7 disable

`disable [--all] [node] [...]`

Configure corosync & pacemaker to not run on node boot on specified node(s), if node is not specified then corosync & pacemaker are disabled on the local node. If `--all` is specified then corosync & pacemaker are disabled on all nodes. (Note: this is the default after installation)

10.5.2.8 standby

`standby [<node>] | --all`

Put specified node into standby mode (the node specified will no longer be able to host resources), if no node or options are specified the current node will be put into standby mode, if `--all` is specified all nodes will be put into standby mode.

10.5.2.9 unstandby

`unstandby [<node>] | --all`

Remove node from standby mode (the node specified will now be able to host resources), if no node or options are specified the current node will be removed from standby mode, if `--all` is specified all nodes will be removed from standby mode.

10.5.2.10 remote-node

`remote-node add <hostname> <resource id> [options]`

Enables the specified resource as a remote-node resource on the specified hostname (hostname should be the same as 'uname -n')

`remote-node remove <hostname>`

Disables any resources configured to be remote-node resource on the specified hostname (hostname should be the same as 'uname -n')

10.5.2.11 status

```
status
```

View current cluster status (an alias of 'pcs status cluster')

10.5.2.12 pcsd-status

```
pcsd-status [node] [...]
```

Get current status of pcsd on nodes specified, or on all nodes configured in corosync.conf if no nodes are specified.

10.5.2.13 certkey

```
certkey <certificate file> <key file>
```

Load custom certificate and key files for use in pcsd.

10.5.2.14 sync

```
sync
```

Synchronize corosync configuration to all nodes found from current corosync.conf file

10.5.2.15 cib

```
cib [filename]
```

Get the raw xml from the CIB (Cluster Information Base). If a filename is provided, we save the cib to that file, otherwise the cib is printed.

10.5.2.16 cib-push

```
cib-push <filename>
```

Push the raw xml from <filename> to the CIB (Cluster Information Base)

10.5.2.17 edit

```
edit
```

Edit the cib in the editor specified by the $EDITOR environment variable and push out any changes upon saving.

10.5.2.18 node add

```
node add <node> [--start] [--enable]
```

Add the node to `corosync.conf` and corosync on all nodes in the cluster and sync the new `corosync.conf` to the new node. If `--start` is specified also start corosync/pacemaker on the new node, if `--enable` is specified enable corosync/pacemaker on new node.

10.5.2.19 node remove

```
node remove <node>
```

Shutdown specified node and remove it from pacemaker and corosync on all other nodes in the cluster.

10.5.2.20 uidgid

```
uidgid
```

List the current configured uids and gids of users allowed to connect to corosync.

10.5.2.21 uidgid add

```
uidgid add [uid=<uid>] [gid=<gid>]
```

Add the specified uid and/or gid to the list of users/groups allowed to connect to corosync.

10.5.2.22 uidgid rm

```
uidgid rm [uid=<uid>] [gid=<gid>]
```

Remove the specified uid and/or gid from the list of users/groups allowed to connect to corosync.

10.5.2.23 corosync

```
corosync <node>
```

Get the `corosync.conf` from the specified node.

10.5.2.24 reload corosync

```
reload corosync
```

Reload the corosync configuration on the current node.

10.5.2.25 destroy

```
destroy [--all]
```

Permanently destroy the cluster on the current node, killing all corosync/pacemaker processes removing all cib files and the `corosync.conf` file. Using '--all' will attempt to destroy the cluster on all nodes configure in the `corosync.conf` file.

WARNING: This command permantly removes any cluster configuration that has been created. It is recommended to run '`pcs cluster stop`' before destroying the cluster.

10.5.2.26 verify

```
verify [-V] [filename]
```

Checks the pacemaker configuration (cib) for syntax and common conceptual errors. If no filename is specified the check is performed on the currently running cluster. If '`-V`' is used more verbose output will be printed

10.5.2.27 report

```
report [--from "YYYY-M-D H:M:S" [--to "YYYY-M-D" H:M:S"]] dest
```

Create a tarball containing everything needed when reporting cluster problems. If '`--from`' and '`--to`' are not used, the report will include the past 24 hours

10.5.3 stonith Subcommand

Use the stonith subcommand to configure fence devices. STONITH stands for “Shoot The Other Node In The Head”.

10.5.3.1 show

```
show [stonith id] [--full]
```

Show all currently configured stonith devices or if a stonith id is specified show the options for the configured stonith device. If `--full` is specified all configured stonith options will be displayed.

10.5.3.2 list

```
list [filter] [--nodesc]
```

Show list of all available stonith agents (if filter is provided then only stonith agents matching the filter will be shown). If `--nodesc` is used then descriptions of stontih agents are not printed.

10.5.3.3 describe

```
describe <stonith agent>
```

Show options for specified stonith agent.

10.5.3.4 create

```
create <stonith id> <stonith device type> [stonith device options]
```

Create stonith device with specified type and options.

10.5.3.5 updated

```
update <stonith id> [stonith device options]
```

Add/Change options to specified stonith id.

10.5.3.6 delete

```
delete <stonith id>
```

Remove stonith id from configuration.

10.5.3.7 cleanup

```
cleanup <stonith id>
```

Cleans up the stonith device in the lrmd (useful to reset the status and failcount). This tells the cluster to forget the operation history of a stonith device and re-detect its current state. This can be useful to purge knowledge of past failures that have since been resolved.

10.5.3.8 level

```
level
```

Lists all of the fencing levels currently configured.

10.5.3.9 level add

```
level add <level> <node> <devices>
```

Add the fencing level for the specified node with a comma-separated list of devices (stonith ids) to attempt for that node at that level. Fence levels are attempted in numerical order (starting with 1) if a level succeeds (meaning all devices are successfully fenced in that level) then no other levels are tried, and the node is considered fenced.

10.5.3.10 level remove

```
level remove <level> [node id] [stonith id] ... [stonith id]
```

Removes the fence level for the level, node and/or devices specified. If no nodes or devices are specified then the fence level is removed.

10.5.3.11 level clear

```
level clear [node|stonith id(s)]
```

Clears the fence levels on the node (or stonith id) specified or clears all fence levels if a node/stonith id is not specified. If more than one stonith id is specified, a comma and no spaces must separate them.

Example: `pcs stonith level clear dev_a,dev_b`

10.5.3.12 level verify

```
level verify
```

Verifies all fence devices and nodes specified in fence levels exist.

10.5.3.13 fence

```
fence <node> [--off]
```

Fence the node specified (if `--off` is specified, use the '`off`' API call to stonith which will turn the node off instead of rebooting it).

10.5.3.14 confirm

```
confirm <node>
```

Confirm that the host specified is currently down

WARNING: if this node is not actually down data corruption/cluster failure can occur.

10.5.4 property

Use the property subcommand to set pacemaker properties.

10.5.4.1 list|show

```
list|show [<property> | --all | --defaults]
```

List property settings (default: lists configured properties) If `--defaults` is specified will show all property defaults, if `--all` is specified, current configured properties will be shown with unset properties and their defaults.

10.5.4.2 set

```
set [--force] [--node <nodename>] <property>=[<value>]
```

Set specific pacemaker properties (if the value is blank then the property is removed from the configuration). If a property is not recognized by pcs the property will not be created unless the '`--force`' is used. If `--node` is used a node attribute is set on the specified node.

10.5.4.3 unset

```
unset [--node <nodename>] <property>
```

Remove property from configuration (or remove attribute from specified node if `--node` is used).

10.5.5 constraint

Use the constraint subcommand to set resource constraints.

10.5.5.1 list|show

```
[list|show] --full
```

List all current location, order and colocation constraints, if `--full` is specified also list the constraint ids.

10.5.5.2 location prefers

```
location <resource id> prefers <node[=score]>...
```

Create a location constraint on a resource to prefer the specified node and score (default score: INFINITY).

10.5.5.3 location avoids

```
location <resource id> avoids <node[=score]>...
```

Create a location constraint on a resource to avoid the specified node and score (default score: INFINITY).

10.5.5.4 location rule

```
location <resource id> rule [role=master|slave] [score=<score>] <expression>
```

Creates a location rule on the specified resource where the expression looks like one of the following:

```
defined|not_defined <attribute>

<attribute> lt|gt|lte|gte|eq|ne <value>

date [start=<start>] [end=<end>] operation=gt|lt|in-range

date-spec <date spec options>...

<expression> and|or <expression>
```

10.5.5.5 location show

```
location show [resources|nodes [node id|resource id]...] [--full]
```

List all the current location constraints, if '`resources`' is specified location constraints are displayed per resource (default), if 'nodes' is specified location constraints are displayed per node. If specific nodes or resources are specified then we only show information about them.

10.5.5.6 location add

```
location add <id> <resource name> <node> <score>
```

Add a location constraint with the appropriate id, resource name, node name and score. (For more advanced pacemaker usage).

10.5.5.7 location remove

```
location remove <id> [<resource name> <node> <score>]
```

Remove a location constraint with the appropriate id, resource name, node name and score. (For more advanced pacemaker usage).

10.5.5.8 order show

```
order show [--full]
```

List all current ordering constraints (if '`--full`' is specified show the internal constraint identifiers as well).

10.5.5.9 order then

```
order [action] <resource id> then [action] <resource id> [options]
```

Add an ordering constraint specifying actions (start,stop,promote, demote) and if no action is specified the default action will be start. Available options are kind=Optional/Mandatory/Serialize and symmetrical=true/false

10.5.5.10 order set

```
order set <resource1> <resource2> [resourceN]... [options] [set
<resourceX> <resourceY> ...]
```

Create an ordered set of resources.

10.5.5.11 order remove

```
order remove <resource1> [resourceN]...
```

Remove resource from any ordering constraint.

10.5.5.12 colocation show

```
colocation show [--full]
```

List all current colocation constraints (if '--full' is specified show the internal constraint identifiers as well).

10.5.5.13 colocation add

```
colocation add [master|slave] <source resource id> with [master|slave]
<target resource id> [score] [options]
```

Request `<source resource>` to run on the same node where pacemaker has determined `<target resource>` should run. Positive values of score mean the resources should be run on the same node, negative values mean the resources should not be run on the same node. Specifying 'INFINITY' (or '-INFINITY') for the score force `<source resource>` to run (or not run) with `<target resource>`. (Score defaults to "INFINITY") A role can be master or slave (if no role is specified, it defaults to 'started').

10.5.5.14 colcation set

```
colocation set <resource1> <resource2> [resourceN]... [setoptions] ...
[set <resourceX> <resourceY> ...] [setoptions  <name>=<value>...]
```

Create a colocation constraint with a resource set.

10.5.5.15 colocation remove

```
colocation remove <source resource id> <target resource id>
```

Remove colocation constraints with `<source resource>`.

10.5.5.16 remove

```
remove [constraint id]...
```

Remove constraint(s) or constraint rules with the specified id(s).

10.5.5.17 ref

```
ref <resource>...
```

List constraints referencing specified resource.

10.5.5.18 rule add

```
rule add <constraint id> [<rule type>] [score=<score>] [id=<rule id>]
<expression|date_expression|date_spec>...
```

Add a rule to a constraint, if score is omitted it defaults to `INFINITY`, if id is omitted one is generated from the constraint id. The `<rule type>` should be '*`expression`*' or '*`date_expression`*'.

10.5.5.19 rule remove

```
rule remove <rule id>
```

Remove a rule if a rule id is specified, if rule is last rule in its constraint, the constraint will be removed.

10.5.6 status

Use the status subcommand to view cluster status.

10.5.6.1 status

View all information about the cluster and resources

10.5.6.2 resources

```
resources
```

View current status of cluster resources.

10.5.6.3 groups

```
groups
```

View currently configured groups and their resources.

10.5.6.4 cluster

```
cluster
```

View current cluster status.

10.5.6.5 corosync

```
corosync
```

View current membership information as seen by corosync.

10.5.6.6 nodes

```
nodes [corosync|both|config]
```

View current status of nodes from pacemaker. If '`corosync`' is specified, print nodes currently configured in corosync, if '`both`' is specified, print nodes from both corosync & pacemaker. If '`config`' is specified, print nodes from corosync & pacemaker configuration.

10.5.6.7 pcsd

```
pcsd <node> ...
```

Show the current status of `pcsd` on the specified nodes

10.5.6.8 xml

```
xml
```

View xml version of status (output from `crm_mon -r -1 -X`).

10.6 Examples

10.6.1 Show all resources

```
# pcs resource show
```

10.6.2 Show options specific to the 'VirtualIP' resource

```
# pcs resource show VirtualIP
```

10.6.3 Create a new resource called 'VirtualIP' with options

```
# pcs resource create VirtualIP ocf:heartbeat:IPaddr2 ip=192.168.0.99
cidr_netmask=32 nic=eth2 op monitor interval=30s
```

10.6.4 Create a new resource called 'VirtualIP' with options

```
#  pcs  resource  create VirtualIP IPaddr2 ip=192.168.0.99
cidr_netmask=32 nic=eth2 op monitor interval=30s
```

10.6.5 Change the ip address of VirtualIP and remove the nic option

```
# pcs resource update VirtualIP ip=192.168.0.98 nic=
```

10.6.6 Delete the VirtualIP resource

```
# pcs resource delete VirtualIP
```

10.6.7 Create the MyStonith stonith fence_virt device to fence host 'f1'

```
# pcs stonith create MyStonith fence_virt pcmk_host_list=f1
```

10.6.8 Disable stonith

```
# pcs property set stonith-enabled=false
```

11 Additional Resources

11.1 HAO Documentation

11.2 IBM z/VM Documentation

1. z/VM Systems Management Application Programming
2. z/VM 6.2 General Information

11.3 Red Hat Documentation

3. RHEL 7 High-Availability Add-On Overview
4. RHEL 7 High-Availability Add-On Administration
5. RHEL 7 High-Availability Add-On Reference
6. RHEL 7 Installation

11.4 Other Documentation

7. Wikipedia – VM Operating System
8. Wikipedia – High Availability Clustering
9. AN!Cluster Tutorial 2
10. About Gluster
11. What is DRBD