Zenoss

The Zenoss Enablement Series:

Zenoss Service Dynamics Resource Manager with Red Hat Cluster Suite Configuration Guide

Document Version 424-P2

Zenoss, Inc.

www.zenoss.com

Table of Contents

Applies To

The procedure outlined in this document applies to the following versions:

- Resource Manager Version 4.2.4 (Build 1859)
- CentOS Linux 6.4
- VMware ESXi 5.0 hypervisor (for cluster nodes)
- Luci version 0.26.0
- Ricci version 0.16.2-23
- DRBD Version 8.4.3

Summary

The objective of setting up the Zenoss Service Dynamics Resource Manager (RM) in a 'high availability' cluster is to minimize, to the greatest degree possible, the downtime associated with a hardware or (non Zenoss) software failure of the server hosting Zenoss. High availability clusters can have various configurations, including, but not limited to:

- Active-Passive, non geo-diverse
- Active-Active non geo-diverse
- Active-Passive, geo-diverse

This document describes an *Active – Passive* high availability cluster without geo diversity that uses Red Hat Cluster Suite (RCHS) and Distributed Replicated Block Device (DRBD). For our scenario, two identical servers per cluster are deployed. At any given time, one node serves as the 'primary' active server and a second identical server stands by ready to take over provision of the key Zenoss services in the event the first server fails or otherwise becomes unavailable. This solution is termed *lacking geo diversity* because the two servers are co-located in the same facility. As such, no protection against a scenario that destroys or renders unreachable the facility hosting Zenoss is provided by this solution.

This manual provides an overview and step-by-step configuration directions to deploy a highly-available Resource Manager on a fast local area network with RHEL or CentOS 6.

Overview

The Resource Manager is composed of the Zenoss Service Dynamics Resource Manager application (RM) and ZenDS - the database server. For large deployments, these services are usually located on separate hosts for better performance. To minimize single point of failure, two clusters are recommended for the Resource Manager - *zenoss* and *ZENDS*.

Understanding Node Roles

Nodes within each cluster assume one of two roles; *active* and *standby*. Each cluster consists of two nodes with only one node assuming the active role at a time. The other node serves as the standby node. The active node responds to requests from users until the service is interrupted, then the roles interchange.

Replicating Data with DRBD

Instead of using shared storage, the cluster replicates data to the passive node through DRBD. DRBD, in this case, eliminates shared storage as a single point of failure.

Using Multiple NICs

Although one network interface card (NIC) can suffice, two NICs are recommended for each member node when fencing is employed. One NIC is used for the public network external requests and heartbeat. The second NIC is used by DRBD for replication service. This method prevents the public network from becoming saturated by the disk replication or synchronization process.

Understanding the VIP

Each cluster has a floating Virtual IP Address (VIP) on the public network. The VIP is assigned to the active node. In case of interruption, the VIP is re-assigned to the standby node.

Managing the Cluster (Remotely)

The cluster can be managed remotely through *luci*, a web based GUI for managing RHCS. All cluster-related configurations can be performed through the GUI. Services can be disabled or restarted through the GUI for performing maintenance tasks. The cluster manager can be installed on a separate machine from the Resource Manager. A single luci instance can be used to manage multiple clusters. For example, a single luci server could be used to manage separate clusters for Resource Manager, Service Impact, and Analytics.

Fencing

Red Hat Clustering requires that a failing node can be disabled to prevent it from affecting the functioning nodes. This is accomplished by a method known as *fencing* that takes advantage of fencing agents included with RHCS. For example, when cluster nodes are deployed onto virtual machines on the VMware vSphere platform, your fencing agent connects to the vSphere API to reboot the failing VM. Conversely, when nodes are hosted on physical servers, those physical servers must be fenced at the hardware level (for example via an out-of-band interface card). The enormous number of fencing and hardware configuration possibilities makes it impractical to document fencing for every possible scenario. Zenoss administrators must work with their IT departments to develop a fencing solution that works with their infrastructure. Although fencing is strongly recommended, administrators who cannot implement a fencing solution can refer to the '"No-Operation" Fencing' section of Appendix B: Fencing for a workaround.

Naming Conventions

The following naming conventions are used in this guide. Replace the example names with the names or values in your environment.

zenoss – cluster name and domain name of the Resource Manager application.
This is a short hostname that will be used by RabbitMQ. It should resolve to the Resource Manager Cluster VIP address (RESOURCEMGRVIP).

ZENDS - cluster name and domain name of the ZenDS.

RESOURCEMGRVIP - virtual IP address of the zenoss cluster.

ZENDSVIP - virtual IP address of the ZENDS cluster.

zenoss{1,2} - general name for the Resource Manager node.

ZENDS{1,2} - general name for the ZenDS node.

zenoss{1,2}-PRIVATE - hostname of the Resource Manager node that resolves to its private IP address. The node should have this name as its hostname, as returned by `uname -n`.

ZENDS{1,2}-PRIVATE - hostname of the ZenDS node that points to its private IP address. The node should have this name as its hostname, as returned by `uname -n`.

zenoss{1,2}-PUBLIC - hostname of the Resource Manager node that points to its public IP address.

ZENDS{1,2 }-PUBLIC- hostname of the ZenDS node that points to its public IP address.

zenoss{1,2}-PUBLICIP - public IP of the Resource Manager node. It is not necessarily a public IP address, but should be reachable by its users.

ZENDS{1,2}-PUBLICIP - public IP of the ZenDS node. It is not necessarily a public IP address, but should be reachable by the users, for example, the Resource Manager server.

zenoss{1,2}-PRIVATEIP - private IP of the Resource Manager node.

ZENDS{1,2}-PRIVATEIP - private IP of the primary ZenDS node.

CLUSTERMGR - hostname of the luci cluster management interface.

The following acronyms are used in this guide:

RM – Zenoss Service Dynamics Resource Manager.

luci – an agent/server architecture for remote administration of clusters.

Sample commands are prepended with prompts that indicate which user issues the command. These prompts include:

- # (pound/hash sign) - execute the command as *root*
- $ (dollar sign) - execute the command as *zenoss*
- `zends>` - execute the command in the zends console

Text in sample commands might be enclosed in less than (**<**) and greater than (**>**) symbols. This indicates the text is a placeholder and the placeholder must be replaced with an actual value for your environment. Examples of text that can include placeholders are *version numbers* and *hostnames*.

Prerequisites

The following hardware requirements must be satisfied to successfully install and configure the Zenoss Resource Manager in a high availability environment:

- One machine for the **luci server**. A single server can be used to manage multiple clusters. Ideally, this machine will have the same architecture as the node systems.
- Two identical RHEL or CentOS machines to function as **Resource Manager nodes**. Consult the *Resource Management Installation* manual for the required system specifications.
- Two identical RHEL or CentOS machines to function as **ZenDS nodes**. Consult the *Resource Management Installation* manual for the required system specifications.
- When fencing is employed, two network interface cards per machine (except luci) with IP addresses configured for both public and private networks.
  When fencing cannot be employed, all cluster traffic should go through a single network interface.
- The same architecture for all node systems. The cluster manager node and cluster nodes should have the same processor architecture (x86_64) and OS version (RHEL or CentOS 6). The cluster manager node configures and manages the clusters and creates DRBD RPM packages. As such, it should share the same architecture as the node systems.
- At least two filesystems per node for the OS and Zenoss data that must be replicated.
- Optionally: a supported fencing device.
  Note: For VMware vSphere, a SOAP license is required. See the RHCS manual for a list of supported devices. See *Appendix B: Fencing* for additional information about fencing.

Consider the following prior to implementing Resource Manager with Red Hat Cluster Suite:

- The host clocks must be synchronized to a time server via Network Time Protocol (NTP).
- SELinux must be disabled on all hosts because it is not supported by Zenoss.
- Nodes should be located within a single LAN with multicast support.
- Nodes must have hostname(s) that resolve to their private IP address. (for example: `resourcemgr1.private.domain.com`)

- There must be a resolvable *hostname* or *domain* name for both private and public IP addresses. If an authoritative DNS server is not available, you can add the entries to the `/etc/hosts` file for each node as shown below. Replace the values with actual IP addresses and hostnames in your environment:

```
RESOURCEMGRVIP            zenoss
zenoss1-PUBLICIP          zenoss1-PUBLIC
zenoss2-PUBLICIP          zenoss2-PUBLIC
zenoss1-PRIVATEIP         zenoss1-PRIVATE
zenoss2-PRIVATEIP         zenoss2-PRIVATE

ZENDSVIP                  zends
ZENDS1-PUBLICIP           ZENDS1-PUBLIC
ZENDS2-PUBLICIP           ZENDS2-PUBLIC
ZENDS1-PRIVATEIP          ZENDS1-PRIVATE
ZENDS2-PRIVATEIP          ZENDS2-PRIVATE
```

Configuration

The following sections describe the installation and configuration tasks that result in a working Zenoss Resource Manager on Red Hat Cluster Suite.

Installing luci on the Controller Server

The cluster manager node is used to configure and manage the clusters. It is also used to create DRBD RPM packages.

Perform the following procedure to install luci as the cluster manager on the *CLUSTERMGR* machine:

1. Update CLUSTERMGR:

   ```
   # yum update
   ```

2. Enter the following commands to ensure the CLUSTERMGR time is synchronized:

   ```
   # chkconfig ntpd on
   # ntpdate pool.ntp.org
   # /etc/init.d/ntpd start
   ```

3. For setup purposes, enter the following commands to disable the internal software firewall:

   ```
   # chkconfig iptables off
   # service iptables stop
   ```

   Note: After you identify the ports for cluster and Zenoss service communications (defined later in this guide), the firewall can be re-enabled with the appropriate ports opened.

4. Reboot the machine:

   ```
   # shutdown -r now
   ```

5. On CLUSTERMGR, install `luci` using yum:

   ```
   # yum install luci
   ```

6. Start `luci`:

   ```
   # service luci start
   ```

7. Configure `luci` to start on boot:

   ```
   # chkconfig --level 12345 luci on
   ```

8. Verify that the cluster hostnames can be resolved from the cluster manager through DNS or the hosts table.

Compiling the DRDB Packages

Perform the following procedure to compile the DRDB packages:

1. On the CLUSTERMGR node, install development tools to enable compiling source code into RPM packages:

```
# yum -y install gcc make automake autoconf flex rpm-build kernel-devel libxslt
```

2. Download the latest version of DRBD from http://oss.linbit.com/drbd/:

```
# cd
# wget http://oss.linbit.com/drbd/<version>/drbd-<version>.tar.gz
```

3. Create the target directories for the RPM packages:

```
# mkdir -p rpmbuild/{BUILD,BUILDROOT,RPMS,SOURCES,SPECS,SRPMS}
```

4. Compile the code with the `--rgmanager` switch:

```
# tar xvfz drbd-<version>.tar.gz
# cd drbd-<version>
# ./configure --with-rgmanager
# make rpm && make km-rpm
```

   Note: The rgmanager switch compiles the resource scripts and other utilities to use DRBD on RHCS.

5. Inspect the *~/rpmbuild/RPMS* directory and verify that the following RPMs were created:
   - drbd-utils
   - drbd-km
   - drbd-rgmanager

   Because some versions of the source code do not include `drbd-rgmanager` in the 'make rpm' process, it might be necessary to build it manually, for example:

```
# cd drbd-<version>
# rpmbuild --bb drbd.spec --with rgmanager
```

   Note: The inclusion of three nodes in each of your clusters is recommended. See the 'Building a Three Node Cluster' section of Appendix B: Fencing for more information. To save on compute resources, the Controller Server can be used as the third node for one of your clusters.

Preparing the ZenDS Nodes

This section describes the required procedures for preparation of the ZenDS nodes. The information in this section has the following caveats:

- Complete all steps in this section and subsections for both ZenDS nodes unless specifically directed.
- Prepare two identical machines for the ZenDS.
- Each machine must have at least two disks (one for the operating system and one for ZenDS data).
- Ensure that public and private hostnames of member nodes are mutually resolvable.

Creating the LVM Disks

Create the LVM disks for each node as follows. The commands in the procedure assume:

- The disk dedicated to the LVM volume is located at `/dev/sdb`.
- The volume group is *zenoss_data*.
- The logical volume is *lv_zends*.

Perform the following procedure to create LVM disks on two nodes:

1. For setup purposes, enter the following commands on all nodes to disable the internal software firewall:

```
# chkconfig iptables off
# service iptables stop
```

   Note: After you identify the communications ports for the cluster and the Zenoss service, you can re-enable the firewall with the appropriate ports opened.

2. Disable SELinux. Because SELinux is not compatible with Zenoss, you must disable it. Enter the following commands on both *Node 1* and *Node 2*:

```
# sed -i 's/SELINUX=enforcing/SELINUX=disabled/' /etc/selinux/config
```

3. Update the nodes:

```
# yum update
```

4. Enter the following commands on both nodes to ensure their times are synchronized:

```
# chkconfig ntpd on
# ntpdate pool.ntp.org
# /etc/init.d/ntpd start
```

5. Reboot each machine:

```
# shutdown -r now
```

6. Issue the following command to create a partition on `/dev/sdb` using the fdisk utility:

```
# fdisk /dev/sdb
```

7. Create the disk partition on `/dev/sdb` and tag the disk as **LVM partition (8e)**. Use the following sequence to create the first LVM partition from the first to last block:

```
n,p,1,<enter>,<enter>,t,8e,w
```

8. Create the *zenoss_data* volume group and *lv_zends* logical disk:

```
# pvcreate /dev/sdb1
# vgcreate zenoss_data /dev/sdb1
# lvcreate -L <size> -n lv_zends zenoss_data
```

Creating the DRBD Resource

Perform the following procedure to create the DRDB Resource for the database nodes:

1. Copy the following rpm files from the cluster manager to the database nodes:
    - drbd-pacemaker-<version>.rpm
    - drbd-utils-<version>.rpm
    - drbd-km-<version>.rpm
    - drbd-rgmanager-<version>.rpm
2. Install the packages and their dependencies:

```
# yum install perl
# rpm -Uhv drbd-utils-<version>.rpm
# rpm -Uhv drbd-km-<version>.rpm
# rpm -Uhv drbd-rgmanager-<version>.rpm
```

3. Copy the following DRDB configuration into the `/etc/drbd.d/global_common.conf` file:

```
global {
    usage-count no;
}
common {
    handlers {
        pri-on-incon-degr "/usr/lib/drbd/notify-pri-on-incon-degr.sh;
/usr/lib/drbd/notify-emergency-reboot.sh; echo b > /proc/sysrq-trigger ; reboot
-f";

        pri-lost-after-sb "/usr/lib/drbd/notify-pri-lost-after-sb.sh;
/usr/lib/drbd/notify-emergency-reboot.sh; echo b > /proc/sysrq-trigger ; reboot
-f";

        local-io-error "/usr/lib/drbd/notify-io-error.sh; /usr/lib/drbd/notify-
emergency-shutdown.sh; echo o > /proc/sysrq-trigger ; halt -f";
    }

    disk {
        on-io-error detach;
        resync-rate 300M;
    }
}
```

Note: The global DRBD configuration shown here intentionally omits fencing directives to prevent multiple fencing. Multiple fencing can cause an infinite reboot loop. Fencing is handled by RHCS rather than DRBD. See *Appendix B: Fencing* for additional information.

4. On both nodes, create the `/etc/drbd.d/r0.res` file.

Note: In the following configuration, substitute *ZENDS{1,2}-PRIVATE* and *ZENDS{1,2}-PRIVATEIP* with the names and IP addresses used in your environment.

Note: The hostname must be consistent with the output of the command '`uname –n`' on all nodes. If it is not, you will encounter the following error:

`r0 not defined in your config (for this host)`

Use the following information for the configuration. When Fencing is not possible, substitute the PUBLICIP value for the single NIC IP being used:

```
resource r0 {
    volume 0 {
        device /dev/drbd0;
        disk /dev/zenoss_data/lv_zends;
        flexible-meta-disk internal;
    }

    net {
        use-rle;
    }

    on ZENDS1-PRIVATE {
        address ZENDS1-PRIVATEIP:7788;
    }

    on ZENDS2-PRIVATE {
        address ZENDS2-PRIVATEIP:7788;
    }
}
```

5. Create the resource `r0`:

```
# drbdadm create-md r0
```

Note: Before starting the DRBD service, verify that the resource on the other node is configured.

6. Start the DRBD service:

```
# service drbd start
# chkconfig --level 12345 drbd on
```

You might need to invalidate one of the nodes before setting one as primary. You can choose any node to invalidate because there is no valid data on the disk yet. Run the command below on the node to be invalidated:

```
# drbdadm invalidate r0
```

7. Set *ZENDS1* as *primary*. Execute the following command on *ZENDS1*:

```
# drbdadm primary r0
```

Allow the nodes enough time to synchronize their disks. Check the status by running the following command:

```
# drbd-overview
```

Continue to the next step when you see the following output from the ‘`drbd-overview`’ command on the primary node:

```
Connected Primary/Secondary UpToDate/UpToDate C
```

8. Initialize the DRDB file system. On *ZENDS1* run the following command:

```
# mkfs -t ext4 /dev/drbd0
```

Note: It is not necessary to perform this initialization on *ZENDS2* because this action is automatically replicated on *ZENDS2*.

9. On both nodes, create the `/opt/zends/data` directory:

```
# mkdir -p /opt/zends/data
```

Note: The directory `/opt/zends/data` is specified instead of `/opt/zends` because the RHCS rgmanager requires the `/opt/zends` directory to be mounted on both nodes to check the status of the zends application.

10. Mount the DRBD disk on *ZENDS1*:

```
# mount /dev/drbd0 /opt/zends/data
```

Note: It is not necessary to add the disk to the `/etc/fstab` file because mounting is managed by the RHCS rgmanager.

Installing ZenDS

The following section describes the procedure to install ZenDS.

Before configuring the ZenDS nodes, verify the following conditions:

- Conflicting messaging systems are removed.
- The DRBD disk is mounted on *ZENDS1*.
- SELinux and the internal software firewall are disabled. The software firewall can be reconfigured once the appropriate ports have been identified and opened.

Perform the following procedure to install ZenDS:

Note: Complete the steps on both ZenDS nodes unless specifically instructed otherwise.

1. Download the latest ZenDS RPM package to each node. Install other dependencies before installing ZenDS. Do not start the `zends` service on either node until instructed to do so below. Issue the following commands to download and install the files:

```
# yum install perl-DBI dmidecode
# rpm -ivh zends-<version>.rpm
```

2. Append the following lines to the `/opt/zends/etc/zends.cnf` file:

```
[mysqld]
open_files_limit=200000
event_scheduler=ON
```

3. On *ZENDS2*, delete the contents of the `/opt/zends/data/` directory:

```
# rm -rf /opt/zends/data/*
```

4. Do **not** start ZENDS on *ZENDS2*. Start ZenDS on *ZENDS1*:

```
# service zends start
```

Note: It is not necessary to start ZenDS on boot because rgmanager handles starting the service.

Configuring the ZenDS Cluster

Perform the following procedure to install required services:

1. On all ZENDS nodes, install *rgmanager*, *ricci*, and *cman*:

```
# yum install rgmanager ricci cman
```

2. Set a password for the user *ricci*. This password will be used by the cluster manager to access to nodes.

```
# passwd ricci
```

3. Configure ricci, cman, rgmanager and modclusterd to start on boot:

```
# chkconfig --level 12345  ricci on
# chkconfig --level 12345  cman on
# chkconfig --level 12345  rgmanager on
# chkconfig --level 12345 modclusterd on
```

4. Start ricci on all ZENDS nodes:

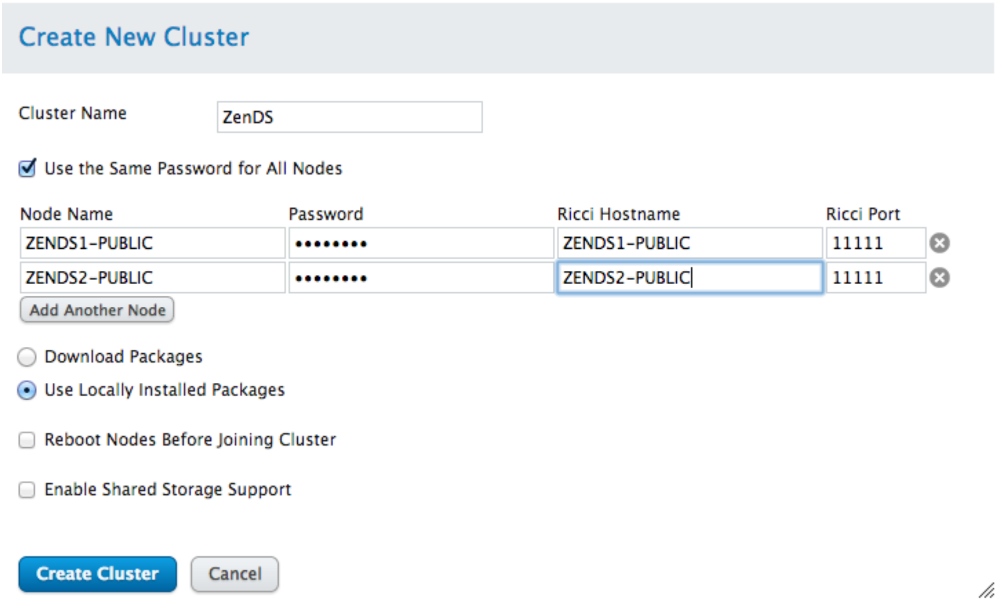
```
# service ricci start
```

5. Browse to *https://CLUSTERMGR:8084* and login to luci as *root*.

Creating the New Cluster

Perform the following procedure to create a new cluster:

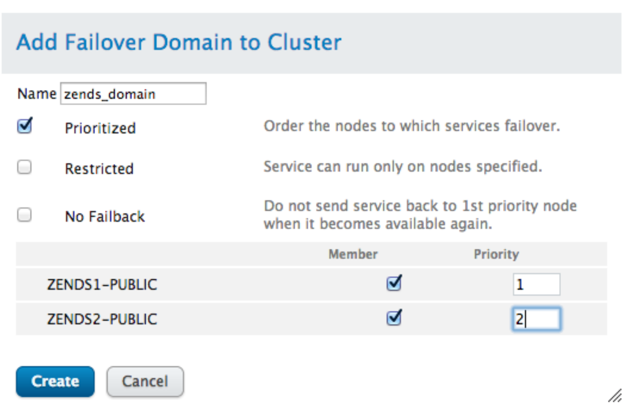
1. Under *Manage Clusters*, click **Create**.
2. Enter **ZenDS** for the *Cluster Name* in this example. Replace *ZenDS* with your cluster name.
3. Enter the *Node Names*, their *Ricci Ports*, *private IP addresses, Users* and *passwords*.
4. Ensure that the node names resolve to the nodes' private IP addresses.
5. Leave other fields as default.
6. Click **Create Cluster**. It might require some time for luci to configure the nodes.
7. If you need to modify a node attribute, click the **Nodes** tab.



Creating a Failover Domain

Perform the following procedure to create a failover domain:

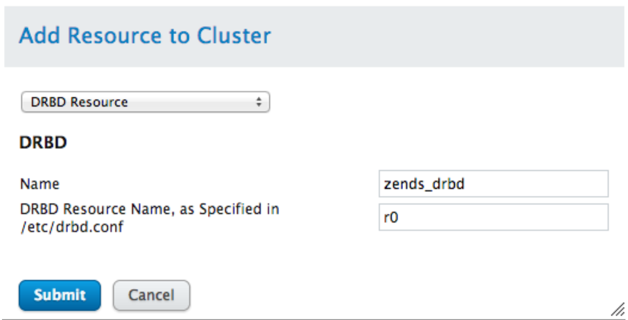
1. Under the *ZenDS* cluster, click the **Failover Domains** tab.
2. Click **Add**.
3. Enter **zends_domain** in the *Name* field.
4. Check **Prioritized**.
5. Uncheck **Restricted**.
6. Uncheck **No Failback**.
7. Check the Member nodes (**ZENDS{1,2}-PUBLIC**).
8. Set the **Priority** of *ZENDS1-PUBLIC* to **1** and *ZENDS2-PUBLIC* to **2**. This means that *ZENDS1* has the higher priority of the two.
9. Click **Create**.



Creating Resources

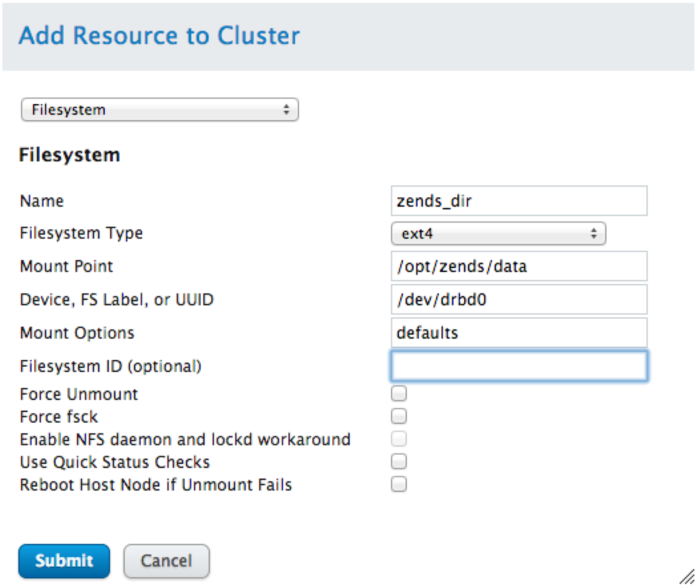
Perform the following procedure to create the required resources:

1. Still under the *ZenDS* cluster, click the **Resources** tab.
2. Click **Add** to create the DRBD resource.
3. Select **DRBD Resource** from the drop-down list as the *resource type*.
4. Enter **zends_drbd** as the *Name.*
5. Enter **r0** as the *DRBD Resource Name*.
   Note: **r0** is the resource name created in the node preparation.
6. Click **Submit**.

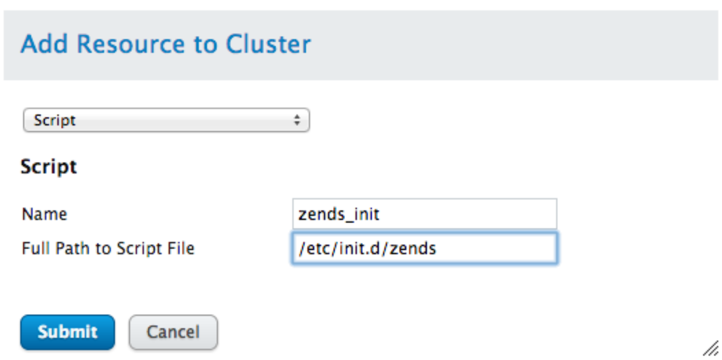


7. Click **Add** to create the *filesystem resource*.
8. Select **Filesystem** as the *resource type*.
9. Enter **zends_dir** as the *Name*.
10. Select **ext4** as the *Filesystem Type*, or enter your file system type if it is different.
11. Enter **/opt/zends/data** as the *Mount Point*.
12. Enter **/dev/drbd0** as the *Device, FS Label, or UUID*.
13. Enter **defaults** as the *Mount Options*.

14. Click **Submit**.

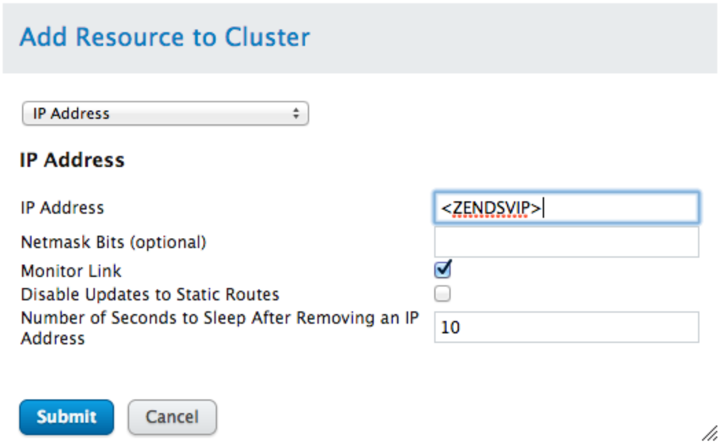


15. Click **Add** to create the *init script resource*.
16. Select **Script** as the *resource type*.
17. Enter **zends_init** as the *Name*.
18. Enter **/etc/init.d/zends** as the *Full Path to Script File*.
19. Click **Submit**.



20. Click **Add** to create the *IP Address resource*.
21. Select **IP Address** as the *resource type*.
22. Enter **ZENDSVIP** as the *IP address* in this example. Replace the value with your VIP address.
23. Verify that **Monitor Link** is checked.

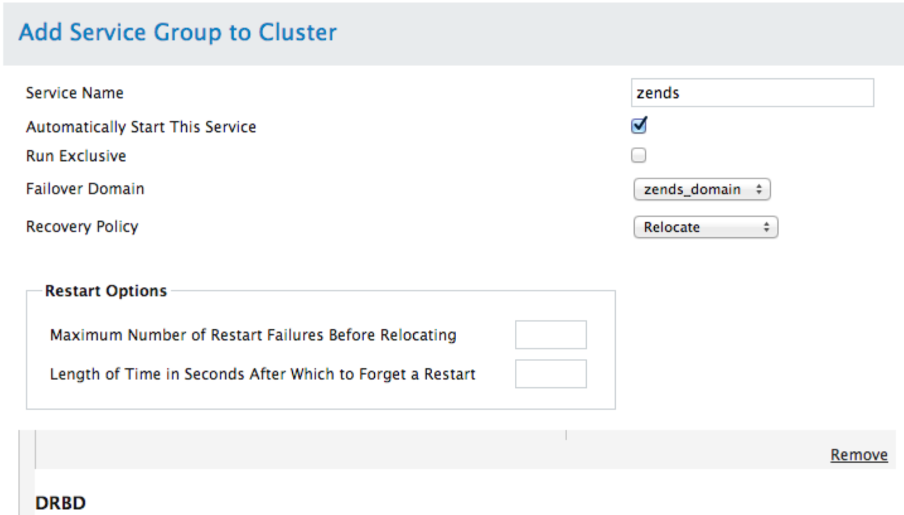
24. Click **Submit**.



Creating Service Groups

Perform the following procedure to create service groups:

1. Still under the *ZenDS* cluster, click the **Service Groups** tab.
2. Click **Add**.
3. Enter **zends** as the *Service Name*.
4. Check **Automatically Start This Service**.
5. Select **zends_domain** as the *Failover Domain*.
6. Select **Relocate** as the *Recovery Policy*.
7. Click **Add Resource**.
8. Select **zends_drbd**.
9. At the bottom of *zends_drbd*, click **Add Child Resource**.
10. Select **zends_dir**.
11. At the bottom of *zends_dir*, click **Add Child Resource**.
12. Select **zends_init**.
13. At the bottom of *zends_init*, click **Add Child Resource**.
14. Select **ZENDSVIP**.
15. Click **Submit**.



Preparing the Resource Manager Nodes

Note: The inclusion of three nodes in each of your clusters is recommended. See the ‘Building a Three Node Cluster’ section of Appendix B: Fencing for more information.

All steps in this section and subsections must be completed for both Resource Manager nodes unless stated otherwise. Prepare two identical machines for Resource Manager with at least two disks (one for the operating system and one for Resource Manager data). Make sure that public and private hostnames of member nodes are resolvable by each other.

Creating the LVM Disks

The following procedure describes how to create the LVM disk for each node. The commands in the procedure assume:

- The disk dedicated to the LVM volume is located on `/dev/sdb`.
- The volume group is *zenoss_data*.
- The logical volumes are *lv_zenoss and lv_zenoss_queue*. The logical volumes are used for Zenoss application data, performance data and the RabbitMQ queue data.

Perform the following procedure to create LVM disks on two nodes:

1. For setup purposes, enter the following commands on all nodes to disable the internal software firewall:

```
# chkconfig iptables off
# service iptables stop
```

   Note: After you identify the communications ports for the cluster and the Zenoss service, you can re-enable the firewall with the appropriate ports opened.

2. Disable SELinux. Because SELinux is not compatible with Zenoss, you must disable it. Enter the following commands on both *zenoss1* and *zenoss2*:

```
# sed -i 's/SELINUX=enforcing/SELINUX=disabled/' /etc/selinux/config
```

3. Update the nodes:

```
# yum update
```

4. Enter the following commands on each node to ensure their times are synchronized:

```
# chkconfig ntpd on
# ntpdate pool.ntp.org
# /etc/init.d/ntpd start
```

5. Reboot each machine:

```
# shutdown -r now
```

6. Issue the following command to create a partition on `/dev/sdb` using the `fdisk` utility:

```
# fdisk /dev/sdb
```

7. Create the disk partition on `/dev/sdb` and tag the disk as **LVM partition (8e)**. Use the following sequence to create the first LVM partition from the first to last block:

```
n,p,1,<enter>,<enter>,t,8e,w
```

8. Create the *zenoss_data* volume group:

```
# pvcreate /dev/sdb1
# vgcreate zenoss_data /dev/sdb1
```

9. Create the logical volumes:

```
# lvcreate -L <size> -n lv_zenoss zenoss_data
# lvcreate -L <size> -n lv_zenoss_queue zenoss_data
```

Creating the DRBD Resource

Perform the following procedure to create the DRDB resource:

1. Copy the following rpm files from the cluster manager to the Zenoss nodes:
   - drbd-utils-<*version*>.rpm
   - drbd-km-<*version*>.rpm
   - drbd-rgmanager-<*version*>.rpm
2. Install the packages:

```
# rpm -Uhv drbd-utils-<version>.rpm
# rpm -Uhv drbd-km-<version>.rpm
# rpm -Uhv drbd-rgmanager-<version>.rpm
```

3. Replace the `/etc/drbd.d/global_common.conf` file with the following configuration:

```
global {
    usage-count no;
}
common {
    handlers {
        pri-on-incon-degr "/usr/lib/drbd/notify-pri-on-incon-degr.sh;
/usr/lib/drbd/notify-emergency-reboot.sh; echo b > /proc/sysrq-trigger ; reboot
-f";

        pri-lost-after-sb "/usr/lib/drbd/notify-pri-lost-after-sb.sh;
/usr/lib/drbd/notify-emergency-reboot.sh; echo b > /proc/sysrq-trigger ; reboot
-f";

        local-io-error "/usr/lib/drbd/notify-io-error.sh; /usr/lib/drbd/notify-
emergency-shutdown.sh; echo o > /proc/sysrq-trigger ; halt -f";
    }

    disk {
        on-io-error detach;
        resync-rate 300M;
    }
}
```

Note: The global DRBD configuration shown here intentionally omits fencing directives to prevent multiple fencing. Multiple fencing can cause an infinite reboot loop. Fencing is handled by RHCS instead of DRBD.

Create the */etc/drbd.d/r0.res* file containing the following configuration. Replace *zenoss{1,2}-PRIVATE* and *zenoss{1,2}-PRIVATEIP* with actual private hostnames and IP addresses, respectively. When Fencing is not possible, substitute the PUBLICIP value for the single NIC being used.

Note: The hostname must be consistent with the output of the command '`uname -n`' on all nodes. If it is not, you will encounter the following error:

```
r0 not defined in your config (for this host)
```

Use the following information for the configuration:

```
resource r0 {
    volume 0 {
        device /dev/drbd0;
        disk /dev/zenoss_data/lv_zenoss;
        flexible-meta-disk internal;
    }
    volume 1 {
        device /dev/drbd1;
        disk /dev/zenoss_data/lv_zenoss_queue;
        flexible-meta-disk internal;
    }

    net {
        use-rle;
    }

    on zenoss1-PRIVATE {
        address zenoss1-PRIVATEIP:7788;
    }

    on zenoss2-PRIVATE {
        address zenoss2-PRIVATEIP:7788;
    }
}
```

4. Create the resource *r0* on both nodes:

```
# drbdadm create-md r0
```

5. Verify that the resource on the other node is configured before starting the DRBD service.
6. Start the DRBD service:

```
# service drbd start
# chkconfig --level 12345 drbd on
```

7. You might need to invalidate a node before specifying one as primary. Because there is no valid data on the disk yet, you can choose any node to invalidate. Issue the following command to invalidate *r0*:

```
# drbdadm invalidate r0
```

Allow the nodes enough time to synchronize their disks. Check the status by running:

```
# drbd-overview
```

Continue to the next step when you see the following output from `drbd-overview` for each of the volumes:

```
Connected Secondary/Secondary UpToDate/UpToDate C
```

8. Make *zenoss1* the primary node. Run the following command on *zenoss1*:

```
# drbdadm primary r0
```

9. On *zenoss1*, initialize the filesystems for the DRBD disks:

```
# mkfs -t ext4 /dev/drbd0
# mkfs -t ext4 /dev/drbd1
```

Note: It is not necessary to initialize the file systems on the second node because the action is replicated automatically.

10. On both nodes, create the `/opt/zenoss` and `/var/lib/rabbitmq/mnesia` mount points:

```
# mkdir -p /opt/zenoss
# mkdir -p /var/lib/rabbitmq/mnesia
```

11. On *zenoss1*, mount **drbd0** on `/opt/zenoss`:

```
# mount /dev/drbd0 /opt/zenoss
```

12. On *zenoss1,* mount **drbd1** to `/var/lib/rabbitmq/mnesia`:

```
# mount /dev/drbd1 /var/lib/rabbitmq/mnesia
```

Installing the Resource Manager Dependencies

Perform the following procedure to install the RM dependencies:

1. On both nodes, confirm that **OpenJDK** is <u>not</u> installed. If it is installed, remove it before proceeding.
2. On both nodes, remove conflicting messaging systems:

```
# rpm -qa | egrep -i "matahari|qpid"
```

3. Install Java SE Runtime Environment version 6:
   a. Download the self-installing RPM of Oracle Java SE Runtime Environment 6u31 from the Java SE 6 Downloads page. The file to download is *jre-6u31-linux-x64-rpm.bin*.
   b. Make the RPM installer executable:

   ```
   # chmod +x /path-to-installer/jre-6u31-linux-x64-rpm.bin
   ```

   c. Start the installer:

   ```
   # /path-to-installer/jre-6u31-linux-x64-rpm.bin
   ```

   d. Add the following line to the end of the */etc/profile* file to append the JAVA_HOME environment variable to it:

   ```
   export JAVA_HOME=/usr/java/default/bin
   ```

   e. Verify the installed version is correct (1.6 Update 31):

```
# java -version
```

4. Install the latest ZenDS package (do not start the service):

```
# yum install perl-DBI dmidecode
# rpm -ivh zends-<version>.rpm
```

5. Prevent zends from running at startup:

```
# chkconfig zends off
```

6. Install the Zenoss dependencies repository on each node:

```
# rpm -ivh http://deps.zenoss.com/yum/zenossdeps-4.2.x-1.el6.noarch.rpm
```

Installing and Configuring RabbitMQ

Use the following procedure to install and configure RabbitMQ:

1. On *zenoss1* and *zenoss2*, install RabbitMQ:

```
# yum -y install rabbitmq-server-2.8.6
```

2. On *zenoss1* and *zenoss2* change ownership of the directories used by RabbitMQ:

```
# chown -R rabbitmq:rabbitmq /var/lib/rabbitmq /var/log/rabbitmq
```

3. On *zenoss1* and *zenoss*2, disable RabbitMQ from starting automatically during boot (RHCS must be used to start RabbitMQ):

```
# chkconfig rabbitmq-server off
```

4. On *zenoss1*, manually start the cluster VIP before starting RabbitMQ. Replace the sample values in this example with values for your environment:

```
# ip addr add <RESOURCEMGRVIP>/32 dev <ethx>
```

5. On *zenoss1* and *zenoss*2, create the `/etc/rabbitmq/rabbitmq-env.conf` file to contain the following content. Replace the sample node name in this example with values for your environment:

   Note: *zenoss* is the short DNS name created in the `/etc/hosts` file that resolves to the VIP.

```
NODENAME="rabbit@<zenoss>"
MNESIA_BASE="/var/lib/rabbitmq/mnesia"
```

6. On *zenoss1*, start the RabbitMQ server daemon:

```
 # service rabbitmq-server start
```

7. On *zenoss1*, update RabbitMQ permissions, using the appropriate zenoss value:

```
# rabbitmqctl -n rabbit@<zenoss> add_user zenoss zenoss
# rabbitmqctl -n rabbit@<zenoss> add_vhost /zenoss
# rabbitmqctl -n rabbit@<zenoss> set_permissions -p \
/zenoss zenoss '.*' '.*' '.*'
```

Installing and Configuring Memcached and Net-SNMP

Use the following procedure to install and configure the `memcached` and `snmpd` daemons:

1. On *zenoss1* and *zenoss2* install Memcached and Net-SNMP:

```
# yum -y install memcached net-snmp net-snmp-utils
```

2. On *zenoss1* only, start the `memcached` daemon:

```
# service memcached start
```

3. On *zenoss1* and *zenoss2* start the `snmpd` daemon:

```
# service snmpd start
```

4. On *zenoss1* and *zenoss2* prevent Memcached from starting automatically on reboot:

```
# chkconfig memcached off
```

5. On *zenoss1* and *zenoss2* configure Net-SNMP to start on boot:

```
# chkconfig snmpd on
```

Installing and Configuring Zenoss

The following sections describe installing and configuring Zenoss Resource Manager on the nodes.

Installing Zenoss Resource Manager

Issue the following command on both *zenoss1* and *zenoss2* to install the Zenoss RPM file:

```
# yum -y --nogpgcheck localinstall zenoss_<version>
```

Installing MySQLTuner

Complete the following procedure on *zenoss1* to download and install the MySQLTuner Perl script:

1. Become the *zenoss* user:

```
# su - zenoss
```

2. Change to the `/opt/zenoss/bin` directory:

```
$ cd /opt/zenoss/bin
```

3. Download the *mysqltuner.pl* script:

```
$ wget mysqltuner.pl
```

4. Specify the appropriate file permissions:

```
$ chmod 755 mysqltuner.pl
```

5. Exit back to the *root* user:

```
$ exit
```

Configuring the Zenoss System

Perform the following procedure to configure Resource Manager:

1. On *zenoss1*, as the *root* user, set the post-installation permissions:

```
# chown -R zenoss:zenoss /opt/zenoss
# chown -R root:zenoss /opt/zenoss/bin/nmap /opt/zenoss/bin/pyraw
/opt/zenoss/bin/zensocket
# chmod 4750 /opt/zenoss/bin/nmap /opt/zenoss/bin/pyraw /opt/zenoss/bin/zensocket
# chown -R rabbitmq:rabbitmq /var/lib/rabbitmq
```

2. On both *zenoss1* and *zenoss2*, prevent Zenoss from starting automatically on reboot:

```
# chkconfig zenoss off
```

3. On *zenoss1*, edit the `/opt/zenoss/etc/global.conf.example` file to refer to the clustered IP or shared node name:

Value to Change	New Value
zodb-host localhost	zodb-host ZENDSVIP
zodb-port 3306	zodb-port 13306
amqphost localhost	amqphost RESOURCEMGRVIP
zep-host localhost	zep-host ZENDSVIP
zep-port 3306	zep-port 13306
zep-uri http://localhost:8084	zep-uri http://RESOURCEMGRVIP:8084

4. On the currently active database node, become the *zenoss* user:

```
# su - zenoss
```

5. Login to Zends to grant IM nodes access to the database:

```
$ zends -u root
```

6. Grant all privileges to the root user on both Resource Manager nodes. Replace the *zenoss1-PUBLICIP* and *zenoss2-PUBLICIP* placeholders with appropriate values for your system:

```
zends> grant all on *.* to 'root'@'zenoss1-PUBLICIP' with grant option;
zends> grant all on *.* to 'root'@'zenoss2-PUBLICIP' with grant option;'
zends> flush privileges;
```

7. On the primary Resource Manage node, as the *zenoss* user verify connectivity to the zends database and troubleshoot if access is denied. The following command should result in an active zends prompt on the database server:

```
$ zends -u root -P 13306 -h ZENDSVIP
```

8. Exit the Zends console.
9. On *zenoss1*, as the *root* user, run the following command to start the system:

```
# service zenoss start
```

Note: Zenoss is configured to write key files during this stage, including creating and indexing the ZODB database. This occurs on the mounted, replicated file system on *zenoss1*. In the background, DRBD replicates the changes to the un-mounted, replicated file systems on *zenoss2*.

Updating the Shared IP

When configuration and startup is complete, on *zenoss1*, update the shared IP for the localhost hub. Doing so ensures that all Zenoss processes can connect to the hub on the VIP. Complete the following steps:

1. Change to the *zenoss* user:

```
# su – zenoss
```

2. Invoke the zendmd interactive console:

```
$ zendmd
```

3. At the zendmd command shell, enter the following commands. Replace *RESOURCEMGRVIP* with your VIP address:

```
>>> dmd.Monitors.Hub.localhost.hostname = "<RESOURCEMGRVIP>"
>>> dmd.Monitors.Hub.localhost._isLocalHost = False
>>> commit()
```

4. Exit the zendmd console.

Local Zenoss Resource Manager Init Scripts

For the cluster manager to successfully fail over a Zenoss Resource Manager instance, the local storage on each cluster member must contain several files that are present in the replicated (DRBD) version of `/opt/zenoss` but are not present on the inactive node's filesystem. These files are required by the Zenoss init script to successfully report on its `stop` and `status` commands:

```
/etc/init.d/zenoss stop
/etc/init.d/zenoss status
```

To meet this need you can perform a one-time `rsync` of the `/opt/zenoss/` directory from the DRBD volume to the local storage on each node after the initial installation. Perform the following commands as the *root* user, unless otherwise instructed:

1. With *zenoss1* as the active cluster node, issue the following command on *zenoss1*:

```
# rsync -avz /opt/zenoss/ zenoss2:/opt/zenoss
```

2. On *zenoss1*, stop Zenoss:

```
# service zenoss stop
```

3. On *zenoss1*, stop `rabbitmq`:

```
# service rabbitmq-server stop
```

4. On *zenoss1*, unmount the *drbd0* filesystem:

```
# umount /opt/zenoss
```

5. On *zenoss1*, unmount the `drbd1` filesystem:

```
# umount /var/lib/rabbitmq/mnesia
```

6. On *zenoss1*, make *zenoss1* the *drbd secondary* for the `r0` resource:

```
# drbdadm secondary r0
```

7. On *zenoss2*, make *zenoss2* the *primary* for the `r0` resource:

```
# drbdadm primary r0
```

8. On *zenoss2*, mount the `drbd0` filesystem:

```
# mount /dev/drdb0 /opt/zenoss
```

9. On *zenoss2*, copy the contents of the drbd volume to *node1*:

```
# rsync -avz /opt/zenoss/ zenoss1:/opt/zenoss
```

10. On *zenoss2*, unmount the /`opt/zenoss` filesystem:

```
# umount /opt/zenoss
```

11. To enable the `stop` command to execute more quickly on the failover target node by only reporting the status of a single daemon, complete the following as the ***zenoss*** user on both *zenoss1* and *zenoss2.*
Note: The drbd volume is intentionally <u>not</u> mounted on either node:

```
$ touch /opt/zenoss/etc/DAEMONS_TXT_ONLY
$ echo zeneventserver > /opt/zenoss/etc/daemons.txt
```

12. On *zenoss2*, (as *root*) make *zenoss2* the *drbd secondary* for the `r0` resource:

```
# drbdadm secondary r0
```

13. On *zenoss1*, make *zenoss1* the *primary* for the `r0` resource:

```
# drbdadm primary r0
```

14. On *zenoss1*, mount **drbd0** on `/opt/zenoss`:

```
# mount /dev/drbd0 /opt/zenoss
```

15. On *zenoss1,* mount **drbd1** on `/var/lib/rabbitmq/mnesia`:

```
# mount /dev/drbd1 /var/lib/rabbitmq/mnesia
```

16. On *zenoss1*, start `rabbitmq`:

```
$ service rabbitmq-server start
```

17. If you want to add monitoring for Windows devices, proceed to the next section.

18. If you do not want to add monitoring for Windows devices using WMI, start `zenoss` on zenoss1 as the *root* user:

```
$ service zenoss start
```

Monitoring Windows Devices (Optional)

If you want to monitor windows devices using WMI, you can optionally install the Windows monitoring rpm. Perform the following procedure on *zenoss1* to install the Windows monitoring rpm:

1. Start the event server and catalog service as the *root* user:

```
# su - zenoss -c "zeneventserver start; zencatalogservice start"
```

2. Install the Windows monitoring rpm:

```
# yum -y --nogpgcheck localinstall zenoss_msmonitor-<version>
```

3. Restart Zenoss:

```
# service zenoss restart
```

4. When Zenoss successfully starts, log on to the user interface of your Zenoss instance. Be sure to access the application by specifying the VIP hostname in the URL. Verify that the application is operating correctly.

Understanding Cluster Functionality

If the application is running properly, the cluster has partial functionality. Zenoss is fully installed and operating on *zenoss1*. It is reading and writing to its files on the mounted, replicating file systems.

Understanding Failover Mounting and Orphaned Files

Zenoss is also installed on the backup node. However, it has written many of its dynamic files, for example, the contents of `/opt/zenoss`, to a duplicate set of directories that will become orphaned when the backup node stands in for the primary node during a failover.

RHCS must be configured to mount the file system replicated by DRBD on the operating system of the failover node when a failure occurs. This is necessary because Zenoss has not been started on the backup node and its version of `/opt/zenoss` does not contain the databases, indexes, or other working files necessary for Zenoss to function. Mounting the replicated file system hides the orphaned copy of `/opt/zenoss` on the backup node. The operating system uses the DRDB copy of the replicated `/opt/zenoss` files that contain all necessary information for Zenoss.

Configuring the Resource Manager Cluster

Perform the following procedure as *root* to configure the RM cluster:

1. On all zenoss nodes, install *rgmanager*, *ricci*, and *cman*:

```
# yum install rgmanager ricci cman
```

2. Set a password for the user *ricci:*

```
# passwd ricci
```

3. Configure ricci, cman, rgmanager and modclusterd to start on boot:

```
# chkconfig --level 12345 ricci on
# chkconfig --level 12345 cman on
# chkconfig --level 12345 rgmanager on
# chkconfig --level 12345 modclusterd on
```

4. Start ricci on all zenoss nodes:

```
# service ricci start
```

5. Browse to **https://CLUSTERMGR:8084** and login to luci as *root*.

Configuring the Cluster

Perform the following procedure to configure the cluster:

1. Under **Manage Clusters**, click **Create**.
2. In the **Cluster Name**, enter *zenoss*.
3. Enter the *Node Names* (public hostname), their *Ricci Ports*, *public IP addresses*, *Users* and *passwords*.
4. Ensure that the node name is a resolvable hostname and resolves to its public IP address.
5. Leave **Other Fields** as *default*.
6. Click **Create**.
7. If you need to modify a node attribute, click the **Nodes** tab.

Creating a Failover Domain

Perform the following procedure to create a failover domain:

1. Under the *zenoss* cluster, click the **Failover Domains** tab.
2. Click **Add**.
3. Enter *resourcemgr_domain* in the *Name Field*.
4. Check **Prioritized**.
5. Check the member nodes, in this example, *zenoss{1,2}-PUBLIC*.
6. Set the **Priority** of *zenoss1-PUBLIC* to **1** and *zenoss2-PUBLIC* to **2**. This means that *zenoss*1-PUBLIC has higher priority of the two.
7. Click **Create**.

Creating Resources

Perform the following procedure to create the required resources:

Adding a DRBD Resource

1. Still under the *zenoss* cluster, click the **Resources** tab.
2. Click **Add** to create the DRBD resource.
3. Select **DRBD Resource** from the drop-down list as the *resource type*.
4. Enter **zenoss_drbd** as the *Name* and **r0** as the *DRBD Resource Name*.
5. Click **Submit**.

Creating The Zenoss File System

1. Click **Add** to create the file system resource.
2. Select **filesystem** from the drop-down list as the *Resource Type*.

3. Enter **zenoss_dir** as the *Name*.
4. Select **ext4** as the *Filesystem Type*, or enter the type of your file system if it is different.
5. Enter **/opt/zenoss** as the *Mount Point*.
6. Enter **/dev/drbd0** as the *Device*.
7. Enter **defaults** as the *Mount Options*.
8. Click **Submit.**

Creating an Additional File System for RabbitMQ

1. Click **Add** to create another file system resource for the RabbitMQ queue.
2. Select **filesystem** from the drop-down list as *Resource Type*.
3. Enter **zenoss_queue_dir** as the *Name*.
4. Select **ext4** as the *Filesystem Type*, or enter the type of your file system if it is different.
5. Enter **/var/lib/rabbitmq/mnesia** as the *Mount Point*.
6. Enter **/dev/drbd1** as the *Device*.
7. Enter **defaults** as the *Mount Options*.
8. Click **Submit**.

Creating Script Resources

1. Click **Add** to create the *init script resource* for the RabbitMQ service.
2. Select **Script** from the drop-down list as the *Resource Type*.
3. Enter **rabbitmq_init** as the *Name*.
4. Enter **/etc/init.d/rabbitmq-server** as the *Full Path to Script File*.
5. Click **Submit**.

1. Click **Add** to create the *init script resource* for the Memcached service.
2. Select **Script** from the drop-down list as the *Resource Type*.
3. Enter **memcached_init** as the *Name*.
4. Enter **/etc/init.d/memcached** as *the Full Path to Script File*.
5. Click **Submit**.

1. Click **Add** to create the *init script resource* for the zenoss service.
2. Select **Script** from the drop-down list as the *Resource Type*.
3. Enter **zenoss_init** as the *Name*.
4. Enter **/etc/init.d/zenoss** as the *Full Path to Script File*.
5. Click **Submit**.

Creating the IP Address Resource

1. Click **Add** to create the *IP Address resource*.
2. Select **IP Address** from the drop-down list as the *Resource Type*.
3. Enter **RESOURCEMGRVIP** as the *IP address*, in this example. Replace this with your VIP address.
4. Ensure that **Monitor Link** is checked.
5. Click **Submit**.

Creating Service Groups

Perform the following procedure to create the service group(s):

1. Still under the zenoss cluster, click **Service Groups** tab.
2. Click **Add**.
3. Enter **resource_manager** as the *Service Name*.
4. Check **Automatically Start this Service**.
5. Select **resourcemgr_domain** as the *Failoverdomain*.
6. Select **Relocate** as the *Recovery Policy*.
7. Click **Add Resource**.
8. Select the **RESOURCEMGRVIP**.

Note: It is important that the VIP gets assigned first to the active node before starting RabbitMQ server.

9. Under *RESOURCEMGRVIP*, click **Add Child Resource.**
10. Under the *RESOURCEMGRVIP* resource, select **memcached_init**.
11. Under *memcached_init*, click **Add Child Resource**.
12. Select **zenoss_drbd**.
13. Under *zenoss_drbd*, click **Add Child Resource**.
14. Select **zenoss_queue_dir**.
15. Under *zenoss_queue_dir*, click **Add Child Resource**.
16. Select **rabbitmq_init**.
17. Under *rabbitmq_init*, click **Add Child Resource**.
18. Select **zenoss_dir**.
19. Under *zenoss_dir*, click **Add Child Resource**.
20. Select **zenoss_init**.
21. Click **Submit**.

Failure Modes

The following sections describe the various failure modes for the system.

Node Failure Types

The node failure types include:

- Active Node Failure
- Passive Node Failure

Active Node Failure

In the case where the active node suffers an outage, the cluster shifts all resources to the other (backup) node. Because the DRBD provides a synchronous replica of the cluster data to the backup node, the cluster can continue to serve from there.

To test node failure scenario, power off the machine and watch check the syslog output. The service should relocate to the backup node.

When the active node resumes, the cluster shifts back to that node because it has the higher failover priority.

Passive Node Failure

If the passive node suffers an outage, no action from the cluster is expected. However, the cluster should recognize the passive node when it comes back online.

Network Failure Types

The network failure types include:

- Public Network Link Failure
- Private Network Link Failure

Public Network Link Failure

Because the cluster monitors the public link, a down link also results in relocation of all resources to the backup node. The scenario can be simulated by unplugging the public interface cable. The result is both nodes will attempt to fence each other. However, only the healthy node can shut down the machine. See *Appendix B: Fencing* for additional information about fencing.

Private Network Link Failure

The private network is only used for DRBD replications. When private network of any node is down, nothing is expected from the cluster. The service should run normally without any failover. DRBD replication will resume replication upon network recovery.

Other Failure Types

The following sections describe the various other failure types, including:

- Storage Subsystem Failure
- Daemon Failure

Storage Subsystem Failure

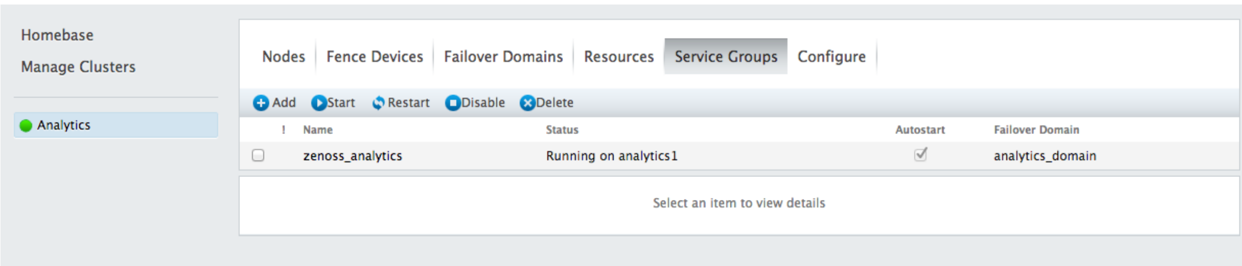
When a node mirrored hard disk fails, the DRBD of the unhealthy node will try to shut down itself. This in turn behaves in a similar way as a node failure. In a VMware environment, simulate this by removing the virtual disk that contains Zenoss data.

Daemon Failure

The active node periodically runs service status command. If the daemon is found to be not running, all resources will be relocated to the backup node. This scenario can be tested by stopping/killing the daemon.

Administration

Clusters can be managed through luci as a cluster manager. The luci server provides a graphical user interface to stop, start, and relocate services. It also enables users to configure the cluster.



Note: Although it is also possible to manage the cluster through the command-line, that discussion is beyond the scope of this document. Consult the RHCS documentation for information about how to manage clusters through the command-line.

Executing Maintenance Tasks on the Cluster

Before performing maintenance task on any cluster member, ensure that the service group is disabled. This is necessary to avoid automatic toggling between nodes. If you must start the service/daemon while performing maintenance, you must start a node manually. Perform the following steps to manually start the daemon:

1. Disable the service group.
2. Choose a node to run the service.
3. Set the *DRBD* resource of that node to *primary*.
4. Mount the disk.
5. Assign the VIP to that node using `ip` command. For example:

```
# ip addr add <RESOURCEMGRVIP>/24 dev eth1
```

6. Start the service.

Appendix A: Known Errors

The following are known errors for this high availability deployment:

- Upon an attempt to start, `cman` reports: `Can't determine address family of nodename`. To troubleshoot, ensure that the node name of the node is resolvable from itself. Add the hostname into the `/etc/hosts` file if you are not using DNS.
- Resources are not relocated.
  Relocation can fail if fencing an unhealthy node is unsuccessful. See *Appendix B: Fencing* for additional information about fencing. Verify the fencing parameters are correct. It is also possible that the secondary node is unable to start a resource (DRBD, IP, etc.). Consult the `/var/log/messages` file for hints.

Appendix B: Fencing

Fencing is an automated means of isolating a node that appears to be malfunctioning to protect the integrity of the DRBD volume(s).

If fencing is required, we recommend placing the fencing device on the public network. The reason for placing the fencing device on the public network is explained by how fencing and communication work in various implementations. In this case the two implementations are a *public* versus a *private* network interface for fencing communications.

Note: Fencing is required by Red Hat. If you do not define a fencing method, your failover will fail with an error "no method defined" when the cluster attempts to fail over to the backup node.

Using a Private Network Interface

Although it is possible to pass heartbeat communications through the private network interface, it requires a complex fencing mechanism. (See the *Implementing/Deploying Quorum Disk on RHCS* manual for more information.) The complex fencing mechanism is prone to issues. Consider, for example, a heartbeat communication that passes through the private network interface. If the private network link fails for either of the nodes, heartbeat communications fail. Each node perceives the other as offline although the active node is still online from the point of view of the user. Because the nodes perceive each other to be off line, each machine initiates fencing of the other node. Both nodes can access the fencing device because the public network links of both nodes are still functioning. The fencing device successfully fences, or shuts down both machines. Fencing both nodes leaves the cluster without a healthy, functioning node online. The complicated work around for this scenario is:

1. Move the fencing device to the private network.
2. Renumber the fencing device.
3. Reconfigure systems that use the fencing device.

Using a Public Network Interface

If heartbeat communications pass through the public network and the public network link for a node goes down, both nodes still try to fence each other. The difference in this case however is that the node with the down public network link cannot communicate with the fencing device. This means that the healthy node can fence the unhealthy node, but the unhealthy node cannot fence the healthy node.

Building a Three Node Cluster

Your cluster will become more robust and resilient if you add a third node to the cluster. The third node need not host resources and can act only as a voting member of the quorum. This means that if the cluster heartbeat link between the nodes is interrupted, one of the nodes will still be able to reach the third member to achieve a cluster quorum vote (>50% of the nodes are required for quorum). The failing node will detect that it is isolated from the other two and does not have a quorum. It will determine that it is no longer eligible to host resources and must shut down any running resources. The Cluster Manager server in this guide can be added to one of your clusters as a non-resource hosting member to save on compute resources.

To prepare the third node for a cluster, perform the following as *root*:

1. Update the node:

```
# yum update
```

2. Enter the following commands to ensure the node's time is synchronized:

```
# chkconfig ntpd on
# ntpdate pool.ntp.org
# /etc/init.d/ntpd start
```

3. For setup purposes, enter the following commands to disable the internal software firewall:

```
# chkconfig iptables off
# service iptables stop
```

   Note: After you identify the ports for cluster communications, the firewall can be re-enabled with the appropriate ports opened.

4. Reboot the machine:

```
# shutdown -r now
```

5. Install `rgmanager` , `ricci` , and `cman` :

```
# yum install rgmanager ricci cman
```

6. Set a password for the user ricci . This password will be used by the cluster manager to access to node.

```
# passwd ricci
```

7. Configure `ricci`, `cman`, `rgmanager` and `modclusterd` to start on boot:

```
# chkconfig --level 12345  ricci on
# chkconfig --level 12345  cman on
# chkconfig --level 12345  rgmanager on
# chkconfig --level 12345 modclusterd on
```

8. Start `ricci` on the node:

```
# service ricci start
```

9. Browse to https://CLUSTERMGR:8084 and login to luci as *root*.

10. To add the third node to a cluster, perform the following:
    a. Navigate to the desired Cluster in the Cluster Manager UI.
    b. Click **Add** to add a new node to the cluster and enter the details for the third node.
    c. Navigate to **Failover Domains** and select your failover domain.
    d. Select the checkbox for **Restricted** to make the cluster resources only run on selected servers.
    e. Make sure that the third node is <u>not</u> selected as a cluster member and does not have a priority.

VMware SOAP Fencing

Perform the following procedure to configure fencing with *VMware SOAP*:

1. Under the *zenoss* cluster in the GUI, click the **Fence Devices** tab.

2. Click **Add**.

3. Select **VMware Fencing (SOAP Interface)**.
4. Enter the name of the VMware server as the *name*.
5. Enter the *hostname* or *IP address*.
6. Enter the *Login* and *Password*.
7. Repeat the above for all Esxi hosts.
8. Navigate to the **Nodes** tab.
9. Select a node.
10. Click the **Add FenceMethod** button.
11. Enter **off** as the *Method Name*.
12. Click **Submit**. This turns off the VM if it was fenced.
13. Click **Add fence device.**
14. Select the fencing device for that node.
15. Enter the UUID of your VM. This can be obtained from the command line of your RHCS node using the following VMware command syntax:

```
./usr/sbin/fence_vmware_soap --ip YourVSphereHost --username root --password -z --
action list | grep YourVMName
```

This returns the VM name, UUID, for example:

```
zenoss_rm,564d9738-9d01-f30f-062a-168023e081be
```

16. Check the **Use SSL** option if the host requires an SSL connection.
17. Repeat steps 8-16 for all nodes.
18. Navigate to the **Nodes** tab.
19. Check all nodes.
20. Click **Join Cluster**. When both nodes have joined, the service group can be started through the GUI.
21. Navigate to the **Service Groups** tab.
22. Check the **resource_manager Service**.
23. Click **Start**. If there are no errors, the zenoss service can be accessed through its VIP address.
24. Browse to the *RESOURCEMANAGERVIP* to verify.

“No-Operation” Fencing

Zenoss strongly suggests that you use a fencing mechanism. In the event that you do not have and cannot obtain any fencing options, you can create a fencing script that can be executed by the cluster manager but performs no actions other than returning `success`. This “No Operation” fencing option bypasses the failure of "no method defined" that will be encountered upon a failover when no fencing is configured. Please note that this is not the recommended configuration, but can be used as a workaround until functional fencing can be configured.

The No-Operation fencing option has the following requirements:

- You must have a cluster with three quorum votes/nodes.
- The cluster heartbeat and drbd traffic must use the same network interface so a node that fails is completely isolated.
- You must manually reboot the failing node to enable it to rejoin the cluster after a failure.

To configure the temporary No-Operation fencing workaround, perform the following:

1. Create a new script on each node:

```
/usr/sbin/noFence
#!/bin/bash
echo "success: noFence"
exit 0
```

2. Make the file executable:

```
# chmod +x /usr/sbin/noFence
```

3. Rename the existing VMware Soap fencing script and replace it with a symbolic link to the new `noFence` script:

```
# mv /usr/sbin/fence_vmware_soap /usr/sbin/fence_vmware_soap.orig
# ln -s /usr/sbin/noFence /usr/sbin/fence_vmware_soap
```

4. Navigate to the desired cluster and click on **Fence Devices**.
5. Click **Add** and create a new *Fence Device* with these details:

```
Type (dropdown menu): VMware Fencing (SOAP Interface)
Name: noFence
IP Address or Hostname: noFence
Login: noFence
Password: noFence
```

6. Add the fence instance to each node in the *Nodes* section of the *Cluster*:
   a. Select the **Node**.
   b. Click **Add Fence Method**.
   c. **Method Name**: *noFence*
   d. **Add Fence Instance**.
   e. Select **Fence Device**: *noFence*
   f. **VM Name**: *noFence*
   g. Repeat for other nodes.

With the No-Op fencing successfully configured, upon failure, when the live node attempts to fence the failing node it immediately succeeds without triggering any actions. These steps should be reversed and functional fencing should be configured as soon as possible.

Appendix C: Handling DRBD Errors

For information about handling DRDB errors, see *The DRBD User's Guide* at http://www.drbd.org/users-guide/. For information on resolving split brain issues specifically, see http://www.drbd.org/users-guide/s-resolve-split-brain.html.