

# Table of Contents

|          |  |           |
|----------|--|-----------|
| <b>1</b> | <b>Preface.....</b>                                  | <b>2</b>  |
| 1.1      | Introduction.....                                    | 2         |
| 1.2      | Intended Audience.....                               | 2         |
| 1.3      | Conventions.....                                     | 2         |
| <b>2</b> | <b>Principles of High Availability.....</b>          | <b>3</b>  |
| 2.1      | Redundancy.....                                      | 3         |
| 2.2      | Availability.....                                    | 3         |
| 2.3      | Client / Server Architecture.....                    | 3         |
| <b>3</b> | <b>Managing Data Redundancy.....</b>                 | <b>5</b>  |
| 3.1      | Shared Storage Architecture.....                     | 5         |
| 3.2      | Replicated Storage Architecture.....                 | 5         |
| <b>4</b> | <b>High Availability With Linuxha.net.....</b>       | <b>7</b>  |
| 4.1      | Definition of Terms.....                             | 7         |
| 4.2      | Data Replication.....                                | 7         |
| 4.3      | Network Configuration.....                           | 8         |
| 4.4      | Data Storage Management.....                         | 8         |
| 4.5      | Linuxha.net Management Directory - /etc/cluster..... | 8         |
| 4.6      | Linuxha.net Configuration File Format.....           | 9         |
| 4.7      | Linuxha.net Commands.....                            | 9         |
| <b>5</b> | <b>Cluster Setup.....</b>                            | <b>11</b> |
| 5.1      | Configuring The Cluster .....                        | 11        |
| 5.2      | Building the Cluster.....                            | 12        |
| 5.3      | Forming the Cluster.....                             | 14        |
| 5.4      | Checking Cluster Status.....                         | 16        |
| 5.5      | Halting The Cluster.....                             | 17        |
| <b>6</b> | <b>Cluster Application Setup.....</b>                | <b>18</b> |
| 6.1      | Installing The Application.....                      | 18        |
| 6.2      | Configuring The Cluster Application .....            | 18        |
| 6.3      | Managing Volume Groups.....                          | 21        |
| 6.4      | Managing Logical Volumes.....                        | 21        |
| 6.5      | Managing File Systems.....                           | 22        |
| 6.6      | Managing Application Data.....                       | 23        |
| 6.7      | Building The Cluster Application.....                | 23        |
| 6.8      | Removing The Cluster Application .....               | 25        |
| 6.9      | Running The Cluster Application.....                 | 25        |
| 6.10     | Halting The Cluster Application.....                 | 25        |
| <b>7</b> | <b>Linuxha.net Cluster Setup Instructions.....</b>   | <b>26</b> |
| 7.1      | Operating System Installation Notes.....             | 26        |
| 7.2      | Network Setup.....                                   | 26        |
| 7.3      | Firewall Configuration.....                          | 28        |
| 7.4      | Disk Configuration.....                              | 28        |
| 7.5      | Secure Shell (SSH) Setup.....                        | 29        |
| 7.6      | Linuxha.net Software Requirements.....               | 30        |
| 7.7      | Linuxha.net Installation.....                        | 31        |
| 7.8      | Building The Cluster.....                            | 32        |
| 7.9      | Forming The Cluster.....                             | 32        |
| 7.10     | Apache Cluster Application.....                      | 33        |
| 7.11     | MySQL Cluster Application.....                       | 35        |
| 7.12     | Samba Cluster Application.....                       | 38        |
| 7.13     | OpenLDAP Cluster Application.....                    | 40        |
| 7.14     | Fedora Directory Server (Fedora-DS).....             | 44        |

# 1 Preface

## 1.1 Introduction

The primary purpose of this guide is to provide step-by-step instructions on setting up a Linuxha.net cluster and applications. This guide also attempts to provide an introduction to:

- Clustering concepts and technologies
- Logical volume management
- Linuxha.net setup and configuration
- Linuxha.net commands
- Troubleshooting Linuxha.net

## 1.2 Intended Audience

This guide is geared towards users unfamiliar with clustering in general and Linuxha.net in particular. The user is expected to have some knowledge of basic Linux operating system commands and an understanding of the Linux file system. The syntax and descriptions of Linux commands used in the administration of Linuxha.net are provided.

## 1.3 Conventions

Commands to be entered at the command prompt or from within a command-line based application will be presented on a light grey background, in a `courier` font. The command line or application prompt will be shown (usually) before the command, the prompt is not to be typed when entering the command. For commands run from the Linux command line, the prompt can be one of three characters:

- `#` – root user prompt, e.g.  

```
# clstat
```
- `$` – regular user prompt, e.g.  

```
$ ls
```
- `>` – line continuation prompt, e.g.  

```
# lvcreate --name data1v \  
> --size 100M \  
> mysqlvg /dev/hda9
```

Command output and text file content will be displayed in `courier` font on a light yellow background, for example:

```
# ps -ef | grep cluster1 | grep -v grep  
root      3291      1  0 10:16 ?        00:00:00 cllockd-cluster1  
root      3295      1  0 10:16 ?        00:00:00 clnetd-cluster1  
root      4740      1  0 11:19 ?        00:00:00 cldaemon-cluster1
```

Important points will be highlighted in bright yellow, for example:

**This command is to be executed on both nodes**

## 2 Principles of High Availability

Before it is possible to design a high availability solution for any application some of the basic principals involved in high availability must be understood. The two main concepts of concern are “redundancy” and “availability”. Highly available environments are typically client/server architectures, where the “server” side is that which is highly available.

### 2.1 Redundancy

Redundancy simply means that there should be more than one component in a system capable of doing the same job. The list below gives an example of which components can be made redundant in a highly available environment:

- [1] More than one network card
- [2] More than one copy of the data
- [3] More than one machine
- [4] More than one location
- [5] More than one power source
- [6] More than one network

Of course, it is usually beyond most budgets to afford all these in a solution. At a minimum the first three requirements should be met.

From this point onwards the term “cluster” will be used to describe the collective hardware used to create the highly available environment. A single machine is referred to interchangeably as a “server” or “node”.

### 2.2 Availability

A highly available system is not one without *any* downtime – it merely means that under most circumstances if there is an outage the affected applications will be made available again to any clients within a designated time period (typically in less than 1 minute, but may be more depending on the complexity of the application).

Note that this means that the failure of the application is usually not transparent – the user typically has to reconnect to the service in question, though much does depend on the intelligence of the client/server architecture of the application in question.

### 2.3 Client / Server Architecture

Usually a high availability cluster consists of one or more applications acting as “services” to clients – such as a web, database or file server. The common theme here is that there is the concept of client / server topology – so any application that can fall into such a category could (in theory) be placed in a cluster.

A cluster can be run in two different “modes”, which are determined by how the applications are configured, rather than a specific cluster configuration. These modes are:

- **Idle Standby**

In this scenario all applications started on a single server. The other server in the cluster hosts no applications under normal circumstances - hence being classed as “idle”.

The advantage of this approach is that it is typically more straightforward to manage - it allows the “idle” node to be easily changed without impacting any running applications. Further since the node is typically not used it is possible to define it as a lower specification server (less memory, CPU's), thus reducing the overall hardware costs for the cluster.

The major disadvantage of this method is hardware utilisation - a standby server is not performing useful work (from the perspective of client software). Thus even though the standby machine might have cost less, it is still essentially providing a low return on Investment.

Idle standby is implicit if only a single application is being hosted by the cluster.

- **Active Standby**

In this scenario the clusters hosts two or more applications. Under normal operating conditions both servers run clustered applications, (though not the same applications).

Since both servers are running applications best use of the available hardware is being made. Of course this does make administration potentially more complex since the administrator cannot assume where applications reside. For example if the administrator uses a script to pass information from the web server to the database server it is not possible to assume that they are running on the same server.

Under these operating conditions both servers are critical, therefore physical loss of a server will result in some applications being migrated. With an idle standby there is only a 50% chance of the loss of the server causing any problems for client access.

The approach taken is entirely determined by the requirements for each installation, since the parameters of the running cluster and applications can be changed at any time as conditions dictate.

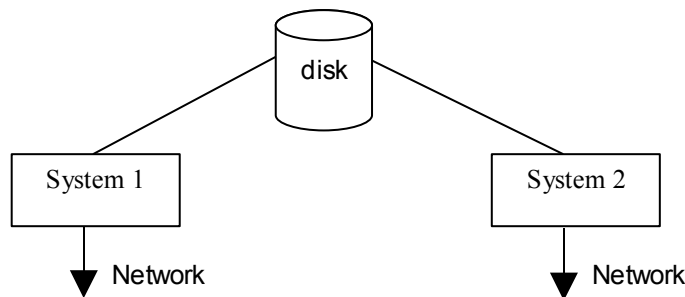
### 3 Managing Data Redundancy

Data availability is the key to making an application highly available. Almost all applications maintain some form of data, such as state, reference or log files, thus the method chosen to ensure the availability of data is critically important.

There are two data storage architectures commonly deployed in a cluster system: Shared Storage and Replicated Storage.

#### 3.1 Shared Storage Architecture

This is the topology used by most commercially available clustering software. In this case the storage used for the application data is stored on a device that is independent of either of the nodes in the cluster. The diagram below shows a typical topology used in commercial clusters:



**Figure 1: Shared Storage Architecture**

- **Advantages**

Relatively simple to configure and manage – there is only a single “copy” of the data..

Since there is only a single data copy the number of nodes that can be attached to the data can be easily increased.

- **Disadvantages**

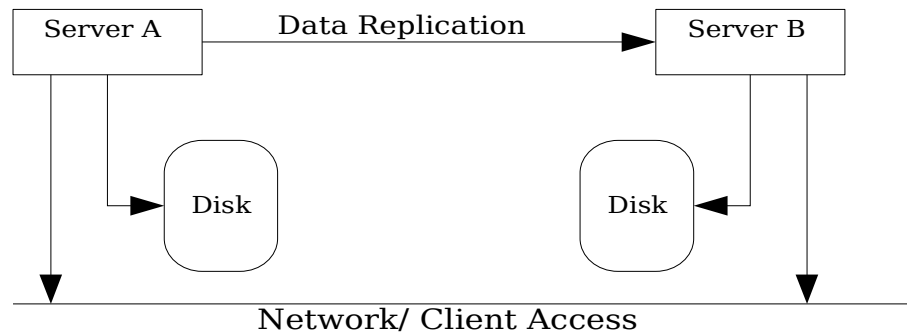
The storage represents a single point of failure. However, this can be overcome with mirroring of some type. Typical commercial environments are likely to use host-based mirrored or even dedicated SAN hardware related infrastructure - such as EMC’s SRDF.

Currently only SCSI or Fibre-channel based storage can be connected to multiple hosts which cost at least twice as much as existing PATA or SATA-based solutions.

#### 3.2 Replicated Storage Architecture

With this topology, each node keeps a “local” copy of the application data, and uses this copy when it serves the application to clients. The cluster can therefore only run an application on a single node at any one time which is the typical requirement for clustering generic applications.

In the scenario shown in Figure 2, when “Server A” is serving the application to clients it will also be sending any updates to the application data to “Server B”.



**Figure 2: Replicated Storage Architecture**

- **Advantages**

Flexibility – the data can be stored on local SCSI, PATA or SATA disks or on iSCSI or Fibre-channel based storage.

Cost effectiveness – a replicated storage solution can cost less than half that of a shared storage solution.

Since replication takes place over a network connection, the nodes can be located anywhere provided that a reliable network connection exists between them.

- **Disadvantages**

The task of keeping two separate copies of data up-to-date can be quite complex.

Data replication may require a large network bandwidth.

## 4 High Availability With Linuxha.net

Clustering with linuxha.net uses a replicated storage architecture, which gives the ability to implement a cluster using commodity hardware, such as PATA or SATA drives. A highly available solution can therefore be deployed at a much lower cost than is currently possible with a shared storage architecture.

### 4.1 Definition of Terms

Before proceeding further, it might be worth reiterating (and refining) a few terms that will be used throughout the remainder of this document:

- **Node**

A node is a machine that is part of the cluster. In the current version of linuxha.net, a cluster can have only two nodes.

- **Application**

This is a discrete service that the cluster provides. The cluster can consist of zero, one or more applications, the maximum is limited only by machine resources.

- **Logical Volume**

A block device which contains a file system and is controlled by a “Logical Volume Manager.(LVM)”. A logical volume (LV) is a virtualization of a disk partition or physical device (PV) that provides additional features suitable for enterprise level applications including on-line resizing and RAID0 striping.

- **Volume Group**

A collection of logical volumes that are made available as a whole or not at all. An application can contain zero or one volume group. A node can have an unlimited number of volume groups.

- **Monitors**

These are programs that check the status of various components, either of an application (such as whether certain processes are running), or the cluster environment (such as network status monitoring).

Most monitors are run as modules of the “Linuxha.net Event Monitoring System (Lems)”. Each running application typically has associated with it a single Lems daemon that makes use of several monitors. These monitors can fall into either of two categories – “system” or “application”.

The system monitors are necessary to monitor the status of the clustered environment from the perspective of a particular application. System monitors typically monitor the IP reachability from the IP addresses for the application and importantly the status of file system data synchronisation. The “application” monitors are typically user-written programs or Perl modules that validate that the current application is still working effectively or not.

- **Service Addresses**

Each application may have associated with it, zero or more IP addresses. It is actually possible to define multiple IP addresses either on the same network, or across multiple networks. Multiple IP addresses on the same network are typically used for applications such as Web servers, whilst IP addresses on multiple networks are typically used to provide “public” and “private” IP access.

### 4.2 Data Replication

A replicated storage architecture requires a mechanism for replicating data over an IP link. This mechanism is referred to as a “network block device” – a device that uses the IP stack for communication to a remote service, whilst providing a standard “block” interface for disk access.

Linuxha.net uses the “DRBD” network block device (see <http://www.drbd.org> for details). DRBD intercepts writes to the local block device, and sends a copy of the data across the network to the DRBD driver on the standby node, so that data on both nodes are in sync. If a node becomes unavailable, DRBD keeps track of

changes to data, so that during resynchronisation only the changes need to be replicated.

Currently, linuxha.net only supports replication over Ethernet.

### 4.3 Network Configuration

Each linuxha.net node requires at least two network cards, with each card being on a separate network. One card is reserved for replication between nodes – the “drbd” or “private network”; the other card(s) are to service network clients – the “public” network. Both networks should be implemented with switches, it is NOT recommended to use a crossover cable for the private network. For redundancy, each network should use a separate switch.

### 4.4 Data Storage Management

As discussed in section 4.1, each application in the cluster can reside on one or more volume groups. The use of Logical Volume Manager (LVM) allows the administrator as well as the Linuxha.net software to validate and manage the environment more easily and is a core component of how replicated data is handled.

Since the Linuxha.net product is built via the provision of duplicated synchronised data resources, the software includes checks to ensure that each volume group that is configured as part of an application is defined identically on both nodes. Such facilities ensure that once the volume groups have been defined on both nodes the software will, if necessary, be able to create and/or validate the volume group configuration for each application created.

Linuxha.net fully supports both LVM version 1 and LVM version 2. It is possible to have different LVM versions running on each node, but this configuration is not recommended.

### 4.5 Linuxha.net Management Directory - /etc/cluster

Linuxha.net stores all files and scripts related to information on the cluster topology, resource allocations, application configuration and current application state in the **/etc/cluster** directory located on both nodes. Some of its content is described in Table 1.

Table 1: Contents of /etc/cluster

| Directory / File                        | Description  |
|---|--|
| /etc/cluster/clconf.xml                 | File that defines the linuxha.net cluster topology. Its format will be discussed in more detail in Section 5.1, further information is also available from its manpage.  |
| /etc/cluster/cllocks.xml                | Configuration file for the linuxha.net lock daemon.  |
| /etc/cluster/.resources                 | Directory which contains several sub-directories that keep track of the resources in use on the particular node.   |
| /etc/cluster/application                | Directory in which the status, resources and definition of the specific clustered application are stored   |
| /etc/cluster/application/TIME           | File which contains the “UNIXTIME” (number of seconds since 00:00 01/01/1970) when the application last ran on this node. This information is used when the contents of <b>/etc/cluster</b> differs on each node. The node which has the most recent time will be assumed to have the most up to date information. |
| /etc/cluster/application/appconf.xml    | File which defines how the cluster application is configured. Its format is described in Section 6.2, further information is also available from its manpage.  |
| /etc/cluster/application/lems.local.xml | Configuration file for the Linuxha Event Management System (Lems) daemon responsible for monitoring the application. Its format is described in Section 6.2  |
| /etc/cluster/application/build          | (Optional) Directory which stores scripts used to initialise the application volume group by creating and populating its logical volumes on both nodes.  |

| Directory / File                 | Description  |
|----------------------------------|--|
| /etc/cluster/application/scripts | (Optional) Directory which contains scripts for starting and stopping the cluster application. |

## 4.6 Linuxha.net Configuration File Format

The configuration of the cluster and its applications is controlled by XML files located in the **/etc/cluster** directory. An XML file consists of one or more sections delimited by the section name in angle brackets, for example the *global* section shown below:

```
<global>
</global>
```

A section can contain subsections and/or elements. An element has a name, zero or more attributes and an optional value and is specified as follows:

```
<element_name
    attribute1="value1"
    attribute2="value2"
    ...
    attributen="valuen">
    value
</element_name>
```

If there is no value, the closing delimiter can be omitted, i.e.

```
<element_name
    attribute1="value1"
    attribute2="value2"
    ...
    attributen="valuen" />
```

For example:

```
<!-- An element with a value and no attributes -->
<key>mykey</key>

<!-- An element with no value and multiple attributes -->
<network name="public" cards="eth0"/>
```

An element is referred to by its name preceded by its parent sections delimited by periods, e.g. **global.name** refers to the **name** element in the **global** section. An element attribute is specified by appending its name to the element name separated by a forward slash, e.g. **node.network/name** refers to the **name** attribute of the **network** element in the **node** section.

Comments can span multiple lines and are delimited by **<!--** and **-->**, for example:

```
<!--
This is a multi-line comment
Line 2
-->
```

## 4.7 Linuxha.net Commands

This section presents an overview of the linuxha.net scripts that are commonly used to manage the cluster and its applications. These commands will be discussed in more detail in Chapters 5 and 6, information is also available in their associated manpages.

Table 2: Linuxha.net Commands

| <i>Command</i>                       | <i>Description</i>  |
|--------------------------------------|---|
| /sbin/cluster/clbuild                | Defines the cluster configuration based on the contents of the <b>/etc/cluster/clconf.xml</b> file.   |
| /sbin/cluster/clform                 | Forms the cluster by starting the cluster daemon on both hosts.   |
| /sbin/cluster/clstat                 | Displays the status of the cluster and its applications.  |
| /sbin/cluster/clhalt                 | Stops the cluster daemon on one or both nodes.  |
| /sbin/cluster/clbuildapp             | Builds or rebuilds a cluster application, it reads application configuration information from <b>/etc/cluster/application/appconf.xml</b> . |
| /sbin/cluster/clrunapp               | Starts the specified cluster application and any other applications on which it depends.  |
| /sbin/cluster/clhaltapp <sup>1</sup> | Stops the specified application.  |
| /sbin/cluster/clremoveapp            | Removes the specified application from the cluster.   |

---

<sup>1</sup>No manpage available

## 5 Cluster Setup

There are three steps to setting up a cluster:

1. Configure
2. Build
3. Form

### 5.1 Configuring The Cluster

The configuration of the cluster is defined by the `/etc/cluster/clconf.xml` file. See Section 4.6 For an explanation of XML file format. To `/etc/cluster/clconf.xml.example` can be copied and modified to create `clconf.xml` when setting up a new cluster.

Listing 1 shows a `clconf.xml` file, elements that may need to be changed or added when setting up a new cluster are bolded and numbered: Their functions are described in Table 3, the numbers can be used to refer to the specific element in the listing.

```
<?xml version="1.0"?>
<clconf>
  <global>
    <name>cluster1</name>                                <!-- 1 -->
    <version>0.1</version>
    <data>replicated</data>
    <datadetail>drbd</datadetail>
    <dataprotocol>C</dataprotocol>
    <logdir>/var/log/cluster</logdir>
    <key>hello_fred</key>                                <!-- 2 -->
    <port>9900</port>
    <clport>9849</clport>
    <clnetdport>9850</clnetdport>
    <portpool>9901,9999</portpool>                       <!-- 3 -->
    <maxblockdevs>50</maxblockdevs>                     <!-- 4 -->
    <drbd_network>a</drbd_network>                       <!-- 5 -->
    <echotype>ICMP</echotype>
  </global>

  <timings>
    <keepalive>3</keepalive>
    <warn>7</warn>
    <dead>12</dead>
    <clusterform>300</clusterform>
  </timings>

  <node>                                                  <!-- 6 -->
    <name>servera</name>                                  <!-- 7 -->
    <network name="a" cards="eth0,eth1"/>                 <!-- 8 -->
    <network name="b" cards="eth2"/>
  </node>

  <node>                                                  <!-- 6 -->
    <name>serverb</name>                                  <!-- 7 -->
    <network name="a" cards="eth0,eth1"/>                 <!-- 8 -->
    <network name="b" cards="eth2"/>
  </node>
</clconf>
```

Listing 1: `clconf.xml`

Table 3: `clconf.xml` Sections and Elements

| Line # | Section / Element | Description  |
|--------|-------------------|--|
| 1      | global.name       | The name of the cluster, value must be alphanumeric with no spaces but can |

| Line # | Section / Element   | Description  |
|--------|---------------------|--|
|        |                     | be any length.   |
| 2      | global.key          | The key which is used to validate all communication between the nodes in the cluster. This must be an alphanumeric string with no spaces or periods.   |
| 3      | global.portpool     | A <i>min,max</i> list that define a range of ports that should be scanned when allocating a port for a replicated file system. The same range of ports must be available on both nodes, and a range covering at least 50 ports should be specified. The number available determines the total number of file systems that the cluster can manage   |
| 4      | global.maxblockdevs | The maximum number of network block devices that can be defined in the cluster. This must provide the same number of devices as the range given in <b>global.portpool</b> , since each network block device requires a network port and a block device.  |
| 5      | global.drbd_network | The logical network name, defined in <b>node/network.name</b> , which is used for communication of DRBD data replication traffic.  |
| 6      | node                | There are two <b>node</b> sections, one for each cluster node.   |
| 7      | node.name           | The name of a cluster node, which must be resolvable to a valid IP address by each node. This address could be on either the public or private interface.  |
| 8      | node.network        | There are two <b>network</b> elements within each <b>node</b> section – one that defines the private (drbd) network, and the other for the public network. The network element has two attributes: <ul style="list-style-type: none"> <li>name – the name assigned to the interface, one of these must be referenced by <b>global.drbd_network</b>.</li> <li>cards – a comma-separated list of operating-system defined interface names, e.g. <i>eth0</i>.</li> </ul> Both nodes must define the same logical networks, although their physical networks ( <b>node.network/cards</b> ) may be different. |

The manpage for **clconf.xml** will provide further information on all its elements.

## 5.2 Building the Cluster

The cluster is created or “built” using the **clbuild** command, it is run on the primary cluster node.

There are two modes in which **clbuild** can be run:

- To build a new cluster run **clbuild** without any arguments, as follows:

```
# clbuild
```

- To rebuild a cluster use:

```
# clbuild --force
```

By default, **clbuild** will only report errors, the **--verbose** option can be use to get a more detailed report.

The **clbuild** command does the following:

- Creates the **/etc/cluster** directory on the secondary node if it does not exist
- Creates DRBD device files on both nodes if necessary.
- Copies **/etc/cluster/clconf.xml** to the secondary node.
- Creates log directories on both nodes if necessary
- Creates resource files in **/etc/cluster/resources** on both nodes, if necessary
- Creates a checksum file for the cluster build.

The output of a successful **clbuild -v** is shown below:

```
INFO 23/12/2005 14:55:32 Checking for required global entries
INFO 23/12/2005 14:55:32 Optional dataprotocol setting validated.
```

```

INFO 23/12/2005 14:55:32 Checking global Port Pool details
INFO 23/12/2005 14:55:32 Checking global Maximum block devices
INFO 23/12/2005 14:55:32 Checking configuration file version information
INFO 23/12/2005 14:55:32 Checking data detail value (and dependent required values)
INFO 23/12/2005 14:55:32 Checking type of data required
INFO 23/12/2005 14:55:32 Global section configuration validation complete
INFO 23/12/2005 14:55:32 Checking cluster timing details
INFO 23/12/2005 14:55:32 Checking node section details
INFO 23/12/2005 14:55:32 Checking IP address resolution
INFO 23/12/2005 14:55:32 Checking ssh cabability between Primary IP addresses (node names).
INFO 23/12/2005 14:55:32 Networks defined for nodel: a,b
INFO 23/12/2005 14:55:32 Networks defined for nodel: a,b
INFO 23/12/2005 14:55:32 Validated network IP for a on nodel: eth0 - 192.100.0.1
INFO 23/12/2005 14:55:32 Validated network IP for b on nodel: eth1 - 192.120.0.1
INFO 23/12/2005 14:55:32 Validated network IP for a on node2: eth0 - 192.100.0.2
INFO 23/12/2005 14:55:33 Validated network IP for b on node2: eth1 - 192.120.0.2
INFO 23/12/2005 14:55:33 Validated unique network/interfaces for nodel:
INFO 23/12/2005 14:55:33 Interface eth0 is on network 192.100.0.0
INFO 23/12/2005 14:55:33 Interface eth1 is on network 192.120.0.0
INFO 23/12/2005 14:55:33 Validated unique network/interfaces for node2:
INFO 23/12/2005 14:55:33 Interface eth0 is on network 192.100.0.0
INFO 23/12/2005 14:55:33 Interface eth1 is on network 192.120.0.0
INFO 23/12/2005 14:55:33 Successfully copied network configuration to centos42s2.
INFO 23/12/2005 14:55:33 Validated data replication network is ok (a).
INFO 23/12/2005 14:55:33 Able to send ping to DRBD Ip address 192.100.0.1 for nodel
INFO 23/12/2005 14:55:34 Able to send ping to DRBD Ip address 192.100.0.2 for node2
INFO 23/12/2005 14:55:34 Node nodel is not running a cldaemon process (good)
INFO 23/12/2005 14:55:34 node node2 is not running a cldaemon process (good)
INFO 23/12/2005 14:55:34 DRBD administration tools found on nodel.
INFO 23/12/2005 14:55:34 DRBD administration tools found on node2.
INFO 23/12/2005 14:55:34 Found LVM v2 on nodel
INFO 23/12/2005 14:55:35 Found LVM v2 on node2
INFO 23/12/2005 14:55:35 LVM v2 command set appears to be installed on nodel
INFO 23/12/2005 14:55:35 LVM v2 command set appears to be installed on node2
INFO 23/12/2005 14:55:35 Physical network check library miitoolib found on nodel.
INFO 23/12/2005 14:55:35 Physical network check library miitoolib found on node2.
INFO 23/12/2005 14:55:35 Creating DRBD devices on nodel...
INFO 23/12/2005 14:55:35 Device entries added to /etc/udev/devices (udev /dev detected).
INFO 23/12/2005 14:55:35 Creating DRBD devices on node2...
INFO 23/12/2005 14:55:35 Device entries added to /etc/udev/devices (udev /dev detected).
INFO 23/12/2005 14:55:36 Validated port allocation files on nodel
INFO 23/12/2005 14:55:36 Validated port allocation files on node2
INFO 23/12/2005 14:55:36 Created 11 DRBD allocation files on nodel
INFO 23/12/2005 14:55:38 Created 11 DRBD allocation files on node2
INFO 23/12/2005 14:55:38 Transferring cluster build checksum to node2
INFO 23/12/2005 14:55:39 Successfully copied clconf.xml to node2
INFO 23/12/2005 14:55:39
INFO 23/12/2005 14:55:39 Clbuild has completed without errors or warnings
INFO 23/12/2005 14:55:39

```

The following are requirements for a successful cluster build:

- **/etc/cluster/clconf.xml** exists on the primary node.
- All mandatory options are present in **clconf.xml**
- Both nodes can communicate with each other using SSH.
- **/etc/cluster/clconf.xml** does not exist on the secondary node, unless the **--force** argument is specified.
- The cluster is not running, unless the **--force** argument is specified.
- The **global.portpool** element in **clconf.xml** has a range of at least ten (10) ports.
- DRBD kernel module exists and can be loaded.
- LVM (either version 1 or 2) exists on both nodes.
- The network names of both nodes (**node/name**) are resolvable to an IP address.
- One of the network names is the same as the local host name.
- The interfaces referenced in **node/network.cards** exist.
- The IP addresses of the interfaces referenced in **node/network.cards** are accessible.
- No interface is referenced more than once in a **node** section.

Table 4 describes common `clbuild` errors and their solutions.

Table 4: `clbuild` Errors

| Error Message   | Solution  |
|---|---|
| Config file <code>/etc/cluster/clconf.xml</code> does not exist   | Copy and edit <code>/etc/cluster/clconf.xml.example</code> to create a new <code>/etc/cluster/clconf.xml</code> .   |
| Node <code>node2</code> already has a <code>clconf.xml</code> – aborting!   | Cluster has already been built, use <code>clbuild --force</code> instead.   |
| Network <code>x</code> defined on <code>node1</code> but not on <code>node2</code> .  | Modify the <b>node</b> sections so that they define the same logical networks, i.e. <b>node/network.name</b>  |
| Data replication network ( <code>drbd_network</code> ) is not defined in configuration.<br>Unknown network <code>x</code> on node <code>node1</code>  | Modify the value of <b>global/drbd_network</b> element in <code>/etc/cluster/clconf.xml</code> so that it matches one of the <b>node/network.name</b> values.     |
| Data replication network ( <code>drbd_network</code> ) is not defined in configuration.<br>Unknown network <code>a</code> on node <code>node1</code>  | Modify the value of the <b>global/drbd_network</b> element in <code>/etc/cluster/clconf.xml</code> so that it matches one of the <b>node/network.name</b> values. |
| Errors while validating IP setting for <code>a</code> on node1:<br>No IP addresses found!   | Correct the invalid interface name specified in <b>node/network.cards</b> .   |
| Duplicated network cards across networks for node <code>node1</code> .<br>Card <code>eth1</code> in <code>b</code> already defined for network <code>a</code> .   | Remove duplicate interfaces from <b>node/network.cards</b> .  |
| Local node not defined in configuration file!<br>When defining a cluster it must consist of configuration information for two nodes - one of which must be the node on which this command is running  | Set the value of one of the <b>node/network.name</b> attributes to be the same as the local host name.  |
| Port pool must cover a range of 10 or more ports.   | Modify the value of <b>global/portpool</b> so that it contains at least 10 ports.   |
| Port pool too small for maximum block devices.  | Modify the value of <b>global/portpool</b> so that the number of ports is at least equal to the number of devices in <b>global/maxblockdevs</b> .                 |
| Unable to resolve IP address of <code>node2</code>  | Add the IP address of <code>node2</code> to <code>/etc/hosts</code> .   |
| An error occurred whilst attempting to communicate from node <code>node2</code> to node <code>node1</code> . Please ensure that the <code>ssh</code> daemon is configured to run on the remote nodes and the <code>ssh</code> binaries client programs exists on both nodes<br>You must also ensure that <code>root:root ssh</code> works without expecting a prompt. | Nodes must be able to communicate using <code>SSH</code> without being prompted for a password.   |

### 5.3 Forming the Cluster

The `clbuild` command only sets up the cluster topology, it is the `clform` command that actually “forms” the cluster by starting the `cldaemon` process on both nodes. The `clform` command can be run on either cluster node.

- Under most circumstances a cluster is formed by executing `clform` without arguments, as follows:

```
# clform
```

- To form a cluster when one of the nodes is unavailable, execute:

```
# clform -force
```

- To join a node to an already running cluster, execute:

```
# clform -join
```

- To form a cluster without starting any “autostart” applications (discussed in Section 6.2), execute:

```
# clform --noapps
```

The output of a successful `clform` command is shown below:

```

INFO 23/12/2005 23:37:14 Validated checksum for cluster configuration
INFO 23/12/2005 23:37:14 SSH communication to node2 will be:
INFO 23/12/2005 23:37:14 192.100.0.2 ("a" network)
INFO 23/12/2005 23:37:14 Checking that the cluster is not already running...
INFO 23/12/2005 23:37:14 *** ATTEMPTING TO FORM CLUSTER cluster1 ***
INFO 23/12/2005 23:37:14 Starting cldaemon on node1...
INFO 23/12/2005 23:37:14 Starting cldaemon on node1...
INFO 23/12/2005 23:37:15 Waiting for cluster to form...
INFO 23/12/2005 23:37:17 Cluster cluster1 started successfully.

```

In addition to displaying its progress on the standard output, `clform` also writes to the `/var/log/cluster/cldaemon-clustername.log` file, where *clustername* refers to the value defined by `global.name` in the `clbuild.xml` file.

The following are requirements for successfully forming a cluster:

- Both nodes can communicate with each other using SSH.
- A successful cluster build
- No changes made to `/etc/cluster/clconf.xml` since the last successful cluster build.
- Time difference between the two nodes is less than ten (10) minutes.

Table 5 describes common `clform` errors and their solutions.

Table 5: *clform* Errors

| Message  | Solution  |
|--|---|
| INFO Validated checksum for cluster configuration<br>INFO Checking that the cluster is not already running...<br>ERROR Cluster cluster1 is already running.  | The cluster has already been formed. If necessary, force the cluster to shut down using <code>clhalt --force</code> , then run <code>clform</code> again.   |
| Cluster has failed to start. Log entries given below:<br>Response to ECHO was FORMING (our state=FORMING)<br>Response to ECHO was STARTING (our state=FORMING)<br>Response to ECHO was UP (our state=FORMING)<br>Both nodes agree UP<br>Time difference between nodes is >10 minutes - require --force to start!   | The time difference between the two nodes is greater than 10 minutes, correct the machine clock if incorrect, otherwise, run <code>clform --force</code>  |
| The cluster configuration file <code>/etc/cluster/clconf.xml</code> appears to have been changed but the changes have not yet been validated.<br>Please run the <code>clbuild(1M)</code> command first before running this command again.<br>Please note that if the cluster is already running you will need to use the <code>--force</code> argument. This will not affect running applications. | The previous <code>clbuild</code> failed because of errors in <code>clconf.xml</code> , or the cluster configuration was changed since the last successful build.<br>Run <code>clhalt --force</code> , correct the errors in <code>/etc/cluster/clconf.xml</code> , if necessary, then run <code>clbuild</code> . |
| Validated checksum for cluster configuration<br>Checking that the cluster is not already running...<br>No remote communication to node2 possible.<br>*** ATTEMPTING TO FORM CLUSTER cluster1 ***<br>Starting cldaemon on node1...<br>Waiting for cluster to form...<br>Cluster did not form in timeout period. Log file entries below:   | One cluster node is unavailable, start the cluster by running <code>clform --force</code> on the working node. Run <code>clform --join</code> on the down node when it becomes available again.   |
| Validated checksum for cluster configuration<br>SSH communication to centos42s2 will be:<br>192.100.0.2 ("a" network)<br>Checking that the cluster is not already running...<br>*** ATTEMPTING TO FORM CLUSTER cluster1 ***<br>Starting cldaemon on node1...<br>Starting cldaemon on node2...<br>Waiting for cluster to form...<br>Cluster did not form in timeout period. Log file entries below: | The firewall must be configured to allow communication between the nodes using SSH. Instructions for doing so are provided in Section 7.3. As a last resort shut down the firewall by running <code>/etc/init.d/iptables stop</code> on both nodes  |

| Message  | Solution |
|--|----------|
| Unable to read from lock daemon.<br>Locking daemon did not start - disabling locking.<br>Network daemon is not running - will attempt to start.<br>Network daemon did not start!<br>Unable to create server on specified socket. |          |

## 5.4 Checking Cluster Status

The cluster status can be determined by checking whether the cluster daemons are running. The name of the cluster daemon processes have the format **daemonname-clustername**, e.g. **cllockd-cluster1**. Execute the following command to check whether daemons for *cluster1* are running:

```
# ps -ef | grep cluster1 | grep -v grep
```

If the cluster is running an output similar to the following is displayed:

```
root      3291      1   0  10:16 ?          00:00:00 cllockd-cluster1
root      3295      1   0  10:16 ?          00:00:00 clnetd-cluster1
root      4740      1   0  11:19 ?          00:00:00 cldaemon-cluster1
```

The **clstat** command can also be used to check the status of the cluster. When run without arguments, it displays: the status of the cluster, the state of each node and each registered application as follows:

- The cluster state - *UP* or *DOWN*.
- The node state:
  - *UP* - the node and its cluster daemon are running
  - *DIED* - the node is running but its cluster daemon is not
  - *DOWN* - both the node and its cluster daemon are shut down.
- The following information about each registered application, or *N/A* if the application is not running:
  - Application - the application name.
  - Node - the name of the node on which the application is running.
  - State - the application state: *STOPPED*, *STARTING*, *STARTED*, *STOPPING*.
  - Started - the length of time (days:hours:minutes) that the application has been running.
  - Monitor - the state of the application monitor, the process responsible for checking the application state. Monitor state can be: *Running*, *Stopped*.
  - Stale - the number of application file systems that are stale, i.e. require synchronisation. An application cannot fail over to another node unless *Stale* is zero.
  - Fail-over - whether the application can fail over to another node or not. If no file system is stale and both nodes are up then *Fail-over* is *Yes*, otherwise *No*.

If an application configuration has changed, **[Rebuild]** is displayed next to the other information indicating that the application needs to be rebuilt (discussed in Section 6.7).

Following is an example of a **clstat** report for a running cluster with two UP nodes:

```
Cluster: cluster1 - UP

      Node      Status
  node1         UP
  node2         UP

Application    Node      State   Started  Monitor  Stale  Fail-over?
  apache       node1    STARTED 0:00:16 Running    0        Yes
  samba        N/A     DOWN    N/A      N/A      N/A      Yes
```

## 5.5 Halting The Cluster

Use the `clhalt` command to halt the cluster or a cluster node, it can be run from either node.

- If the cluster has no running applications, it can be halted as follows:

```
# clhalt
```

- While not the recommended procedure, it is possible to force the applications to shut down when halting the cluster by executing:

```
# clhalt --force
```

- To shut down a single node (*node1*), and automatically restart its applications on the other node, use:

```
# clhalt --node node1 --action failover
```

The output of a successful `clhalt` is shown below:

```
Validated checksum for cluster configuration.
INFO 24/12/2005 11:04:33 Attempting to halt cluster cluster1...
INFO 24/12/2005 11:04:33 Attempting to contact a cluster daemon...
INFO 24/12/2005 11:04:33 Connecting to cluster daemon via host node1
INFO 24/12/2005 11:04:33 Asking cluster daemons to abort...
INFO 24/12/2005 11:04:36 Cluster daemons aborted - cluster cluster1 is DOWN.
```

Table 6 describes some common `clhalt` errors and their solutions.

Table 6: `clhalt` Errors

| Error Message  | Solution   |
|--|--|
| Validated checksum for cluster configuration.<br>Attempting to halt cluster cluster1...<br>Attempting to contact a cluster daemon...<br>SSH communication to node2 will be:<br>192.100.0.2 ("a" network)<br>Cluster already appears to be down.  | Do nothing, cluster has already been halted  |
| Validated checksum for cluster configuration.<br>Attempting to halt cluster cluster1...<br>Attempting to contact a cluster daemon...<br>Connecting to cluster daemon via host centos42s1<br>Unable to stop cluster - 1 application is still running  | Use <code>clhaltapp</code> (described in Section 6.10) to halt running applications then run <code>clhalt</code> , or run <code>clhalt --force</code> to force the cluster and all running applications to halt. |
| The cluster configuration file <code>/etc/cluster/clconf.xml</code> appears to have been changed but the changes have not yet been validated.<br>Please run the <code>clbuild(1M)</code> command first before running this command again.<br>Please note that if the cluster is already running you will need to use the <code>--force</code> argument. This will not affect running applications.<br><code>clhalt</code> exiting with error code 6. | Force the cluster to shut down by executing <code>clhalt --force --nochecksums</code> . Rebuild cluster before restarting it.  |

## 6 Cluster Application Setup

There are eight (8) steps to setting up a cluster application:

1. Install application
2. Configure application
3. Create application volume group
4. Create logical volumes
5. Create and mount file systems.
6. Populate file systems.
7. Build application.
8. Run application.

### 6.1 Installing The Application

There are basically three (3) ways that an application can be installed on a Linux server.

1. Selecting the application during operating system installation.
2. Using the `yum` command to install the application from the yum cache (`/var/yum/cache`) or from a repository URL, e. g.:

```
# yum install package
```

3. Using the `rpm` command to install the application from an rpm file, e.g.:

```
# rpm --install package.rpm
```

### 6.2 Configuring The Cluster Application

Each application stores its configuration scripts in the `/etc/cluster/application` directory, where *application* is the name of the application, e.g. `/etc/cluster/apache`. The `/etc/cluster/application/appconf.xml` file is used to configure the cluster application. The file uses the XML format discussed in Section 4.6.

Linuxha.net provides a sample application configuration file – `/etc/cluster/appconf.xml.example` – which can be copied and modified to create an `appconf.xml` file for a new cluster application.

Listings 2 shows an `appconf.xml` file, the elements and attributes which may need to added or modified when configuring a new application are highlighted and numbered. Table 7 describes these elements and attributes. The numbers in the first column refer to the line numbers in the listing.

Table 7: `appconf.xml` Elements and Attributes

| Line # | Element / Attribute      | Description   |
|--------|--------------------------|---|
| 1      | global.name              | Application name – value can be any length and can contain underscores and any alphanumeric character except spaces.  |
| 2      | global.autostart         | (Optional) Whether the application starts automatically when the cluster is formed or not – value can be <i>yes</i> or <i>no</i> , indicating. If omitted, it defaults to <i>no</i> .   |
| 3      | global.dependencies      | (Optional) Comma-separated list of applications on which current application depends.   |
| 4      | networks.network/net     | Logical name of the network on which the application is running, must refer to one of the <b>node network/name</b> values in <code>clconf.xml</code> (see Section 5.1). It is recommended that the application run on the public network. |
| 5      | networks.network/ip      | A comma-separated list of one or more IP addresses on which the application is running. The addresses cannot be bound to a network card.  |
| 6      | networks.network/netmask | The netmask to use when assigning the IP address.   |

| Line # | Element / Attribute           | Description  |
|--------|-------------------------------|--|
| 7      | networks.network/checklist    | (Optional) Comma-separated list of IP addresses that should be pinged to provide IP level network functionality testing.                                     |
| 8      | networks.network/checkpercent | (Optional) The percentage of successful responses to a ping for the address in networks.network/checklist to be considered viable - value between 1 and 100. |
| 9      | vg.name                       | Name of application volume group   |
| 10     | application.startscript       | Script to start the application.   |
| 11     | application.stopscript        | Script to stop the application.  |

```

<?xml version="1.0"?>
<appconf>
  <global>
    <version>0.3</version>
    <name>app01</name>                                <!-- 1 -->
    <takeover>normal</takeover>
    <syncrate>2048</syncrate>
    <preferred_node>LEAST_CPU_LOAD</preferred_node>
    <autostart>no</autostart>                          <!-- 2 -->
    <dependencies></dependencies>                     <!-- 3 -->
  </global>

  <networks>
    <network
      net="main"                                       <!-- 4 -->
      ip="172.16.177.200"                             <!-- 5 -->
      netmask="255.255.255.0"                         <!-- 6 -->
      checklist="172.16.177.1"                       <!-- 7 -->
      checkpercent="100"                              <!-- 8 -->
      pingtype="icmp"
      pingtimeout="3"/>
    </networks>

  <vg>
    <name>app01vg</name>                              <!-- 9 -->
    <type>filesystems</type>
  </vg>

  <application>
    <startscript>startapp</startscript>              <!-- 10 -->
    <stopscript>stopapp</stopscript>                 <!-- 11 -->
    <maxstoptime>10</maxstoptime>
    <maxstarttime>20</maxstarttime>
  </application>
</appconf>

```

Listing 2: appconf.xml

Additional information on the **appconf.xml** elements and attributes can be found in its manpage.

Each cluster application can have associated with it a process, known as a Linuxha Event Management System (Lems) process, which is responsible for monitoring the application state and taking actions based on state changes. The **/etc/cluster/application/lems.local.xml** file is used to configure the application-specific Lems process. When configuring a new cluster application, the **/etc/cluster/lems.xml.example** file can be copied and modified to create **lems.local.xml**.

The **lems.local.xml** file consists of a **globals** section followed by one or more **check** sections. The **check** sections define the different checks the Lems process carries out. Listing 3 displays an extract of a **lems.local.xml** file, Table 8 describes its elements and attributes.

Listing 3: lems.local.xml

```

<?xml version="1.0"?>
<lems_config>
  <globals modules="/sbin/cluster/lems/modules"
    programs="/sbin/cluster/lems/programs"
    logs="/var/log/cluster/lems"
    port="8800"
  />

  <check>
    <name>flag_check</name>
    <type>internal</type>
    <module>flag_check apache</module>
    <interval>5</interval>
    <action_list>
      <action rc="0" action="NOP"/>
      <action rc="1" action="%RCDATA%"/>
      <action rc="2" action="ABORT"/>
    </action_list>
  </check>

  <!-- more check sections -->

</lems_config>

```

Table 8: lems.local.xml Elements and Attributes

| Element / Attribute             | Description   |
|---------------------------------|---|
| globals.modules                 | Directory where Lems modules are stored   |
| globals.program                 | Directory where Lems programs are stored  |
| globals.logs                    | Directory where Lems logs are stored  |
| globals.port                    | Port on which Lems process runs, each process must run on a different port.   |
| check.name                      | The name of the check   |
| check.type                      | The type of check, value can be either <i>internal</i> or <i>program</i> .  |
| check.module                    | The name of the module or program that executes the check, along with any arguments to the module. Available modules are: <ul style="list-style-type: none"> <li><i>flag_check</i> – stops and starts other modules based on the presence of files in the <b>/etc/cluster/application/flags</b> directory</li> <li><i>ip_module</i> – monitors the application IP address</li> <li><i>link_module</i> – monitors network interface state</li> <li><i>ip_move_interface</i> – moves the application IP address to a different interface if current interface fails</li> <li><i>fsmon</i> – handles consistency of data between nodes</li> <li><i>procmon</i> – monitors application process</li> <li><i>swap_check</i> – monitors free swap space</li> <li><i>capacity_check</i> – monitors free disk space</li> </ul> |
| check.interval                  | The minimum amount of time (in seconds) to wait before running the check for this monitor. Can be any numeric value greater than 0.5.   |
| check.action_list               | Section that contains one or more <b>action</b> elements which define the status codes returned by the check module and the actions to take in response.  |
| check.action_list.action/rc     | The status code returned by the check module  |
| check.action_list.action/action | Action to take in response, can be one of the following: <ul style="list-style-type: none"> <li><i>NOP</i> – do nothing</li> <li><i>HALT</i> or <i>STOP</i> – suspend specified check</li> <li><i>START</i> or <i>RUN</i> – start specified check</li> <li><i>ABORT</i> – abort Lems scheduler</li> </ul>   |

| Element / Attribute | Description   |
|---------------------|---|
|                     | <ul style="list-style-type: none"> <li>• <i>FAILOVER</i> – failover application to other cluster node</li> <li>• <i>HALTAPP</i> or <i>STOPAPP</i> – halt application without failover</li> <li>• <i>PAUSE</i> – suspend check for specified number of seconds</li> <li>• <i>RUNCMD</i> – run specified command</li> <li>• <i>REMOVEMON</i> – remove Lems monitor</li> </ul> |

### 6.3 Managing Volume Groups

As discussed in Section 4.1, Linuxha.net stores its data on logical volume groups rather than on disk partitions. Logical volumes are managed by the Logical Volume Manager (LVM) and support such enterprise features as on-line file system expansion.

Each cluster application has associated with it one volume group. A volume group can consist of one or more disk partitions (physical volumes). Disk partitions are created using the `fdisk` command (discussed in Section 7.4). There are two steps to creating a volume group:

1. Initialise the physical volume(s) using the `pvcreate` command, which has the following syntax:

```
# pvcreate partition [partition ...]
```

2. Create a volume group consisting of the physical volumes created in step 1 using the `vgcreate` command, which has the following syntax:

```
# vgcreate volumegroupname physicalvolume [physicalvolume ...]
```

The application volume group has to be created on both nodes prior to building the application.

Table 9 describes some additional volume group management commands.

Table 9: Volume Group Management Commands

| Command   | Description   |
|---|---|
| <code>pvs [physicalvolume ...]</code>                     | Displays summary information about specified physical volumes, or all physical volumes if none is specified.          |
| <code>pvdisplay [physicalvolume ...]</code>               | Displays detailed information about specified physical volumes, or all physical volumes if none is specified.         |
| <code>pvremove physicalvolume [physicalvolume ...]</code> | Removes physical volume label from partition so that it is no longer recognised by the LVM, partition is not removed. |
| <code>vgs [volume group ...]</code>                       | Displays summary information about specified volume groups, or all volume groups if none is specified.                |
| <code>vgdisplay [volume group ...]</code>                 | Displays detailed information about specified volume groups, or all volume groups if none is specified.               |
| <code>vgremove volume group [volume group ...]</code>     | Removes specified volume groups if they contain no logical volumes.   |

Note: When a volume group management command (e.g. `vgcreate`) is executed on a node which is running Linuxha.net cluster applications, messages similar to the following may appear – they can be safely ignored!

```
/dev/drbd15: read failed after 0 of 4096 at 268369920: Input/output error
/dev/drbd15: read failed after 0 of 4096 at 0: Input/output error
```

### 6.4 Managing Logical Volumes

A volume group can be subdivided into one or more logical volumes, each logical volume contains a file system. The number of logical volumes created depends upon the application requirements. It may be useful in some situations to separate data files and log files, in which case each set of files can be stored on a separate logical volume.

The `lvcreate` command is used to create a logical volume, it has the following syntax.

```
# lvcreate \
  --name logicalvolumename \
  --size volumesize[kMGT] \
  volumegroup \
  partition [partition ...]
```

Append k, M, G or T to *volumesize* to specify units in kilobytes, megabytes, etc., default is M.

The logical volume(s) must be created on the primary node before building the application. The build process can create the logical volumes on the secondary node, however, the process is not affected if the logical volume(s) are built beforehand.

Table 10 describes other logical volume management commands.

Table 10: Logical Volume Management Commands

| Command   | Description  |
|---|--|
| <code>lvs</code>  | Displays summary information about specified logical volume, or all logical volumes if none is specified.                        |
| <code>lvdisplay [volume]group</code>  | Displays detailed information about specified logical volumes, or all logical volumes if none is specified.                      |
| <code>lvremove <i>logicalvolumepath</i> [<i>logicalvolumepath ...</i>]</code> | Removes specified logical volumes, <i>logicalvolumepath</i> is in the format <b><i>/dev/volume</i>group/<i>logicalvolume</i></b> |
| <code>lvremove <i>volume</i>group [<i>volume</i>group ...]</code>             | Removes all logical volumes from specified volume groups.  |

## 6.5 Managing File Systems

A file system has to be created on the logical volume before it can be used. Linuxha.net places no restriction on file system type, although use of a journaling file system is recommended because they do not need to be checked after a system crash and therefore allow faster recovery .

Table 11 describes journaling file systems supported by DRBD.

Table 11: Supported Journaling File Systems

| File System | Description   |
|-------------|---|
| ext3        | <ul style="list-style-type: none"> <li>Default file system for the Red Hat Linux, Fedora, Debian and Ubuntu Linux distributions</li> <li>Not as fast or scalable as reiserfs or xfs.</li> </ul>   |
| jfs         | <ul style="list-style-type: none"> <li>Linux version of the IBM Journaling File System.</li> <li>Faster than reiserfs for small file system sizes (&lt;50 MB)</li> <li>Can be downloaded from <a href="http://freshmeat.net/redir/jfs/4819/url_tgz/jfsutils-1.1.10.tar.gz">http://freshmeat.net/redir/jfs/4819/url_tgz/jfsutils-1.1.10.tar.gz</a>.</li> </ul> |
| reiserfs    | <ul style="list-style-type: none"> <li>Default file system for the Slackware, SuSE, Xandros, and Linspire Linux distributions.</li> <li>Faster than ext3 or jfs</li> </ul>  |
| xfs         | <ul style="list-style-type: none"> <li>Available with the SuSE, Gentoo, Mandrake, Fedora, Slackware and Debian Linux distributions.</li> <li>Some issues have been reported with xfs and DRBD 0.7.</li> </ul>   |

The `mkfs` command is used to create file systems, it has the following syntax:

```
# mkfs -t fstype device
```

where *fstype* is the file system type, e.g. ext3; and *device* is the device path, for a logical volume the path format is ***/dev/volume*group/*logicalvolume***.

In order for the file system to be available for use it has to be mounted. The `mount` command is used, and has the following syntax:

```
# mount -t fstype device mountpoint
```

where *fstype* is the file system type; *device* is the logical volume device path, and *mountpoint* is an existing directory on which the file system is to be mounted.

**Note:** The volume group file systems have to be mounted on the primary node prior to the application build process.

## 6.6 Managing Application Data

In order for the application to be managed by Linuxha.net, its data must be stored on a volume group file system. It may be necessary to also store other types of application files, such as configuration files and log files, on the volume groups. The default application installation would have placed these files in specific directories, these directories therefore have to be copied to the volume group file systems.

There is no straightforward method for determining which application files are to be used to populate the volume group file systems. However there are some commands that can be used to identify application files, described in Table 12.

Table 12: Commands to Display Application File Names

| Command   | Description   |
|---|---|
| <code>find / -name <i>applicationname</i> -print</code>       | Locate files related to a particular application. Wild cards can be specified in <i>applicationname</i> , in which case it must be enclosed in double-quotes. |
| <code>rpm --query --configfiles <i>applicationname</i></code> | List the configuration files associated with application, if it was installed using the <code>rpm</code> utility.   |
| <code>rpm --query --list <i>applicationname</i></code>        | List all files associated with application, if it was installed using the <code>rpm</code> utility.   |

Before populating the application volume groups, the application must be shut down. The shutdown script is specific to the application but usually has the following format:

```
# /etc/init.d/daemonname stop
```

where *daemonname* is the name of the process that controls the application.

## 6.7 Building The Cluster Application

The `clbuildapp` command is used to build or register a cluster application. It can be executed whether the cluster is running or not. There are four (4) stages to the build process, each stage is initiated by running `clbuildapp` with a different argument as follows:

1. Check application configuration

```
# clbuildapp --verbose --check --application appname
```

2. Validate and build logical volumes on both nodes.

```
# clbuildapp --verbose --vgbuild --application appname
```

3. Allocate application resources

```
# clbuildapp --verbose --build --application appname
```

4. Synchronise volume group contents on both nodes.

```
# clbuildapp --verbose --sync --application appname
```

where *appname* is the name of the application.

If the `--verbose` argument is supplied, `clbuildapp` displays a detailed status report while it is running, otherwise only error messages are displayed.

If changes are made to the application configuration, the application has to be rebuilt by rerunning the four stages as follows:

1. Check application configuration

```
# clbuildapp --verbose --check --application appname
```

2. Validate and rebuild logical volumes on both nodes.

```
# clbuildapp --verbose --vgbuild --application appname
```

3. Allocate application resources

```
# clbuildapp --verbose --build --application appname --force
```

4. Synchronise volume group contents on both nodes.

```
# clbuildapp --verbose --sync --application appname --force --forcesync
```

where *appname* is the name of the application.

Table 13 describes common `clbuildapp` errors and their solutions.

Table 13: *clbuildapp* Errors

| Error Message  | Solution   |
|--|--|
| <i>Stage 1: Check</i>  |  |
| Network d unknown to cluster topology.   | Value of <b>networks.network/net</b> in <b>appconf.xml</b> must refer to a network defined in <b>node</b> section of <b>clconf.xml</b> .   |
| Changing the cluster configuration whilst the application is currently running requires use of <code>--force</code> .  | Shut down application using <code>clhaltapp</code><br>OR<br>Rerun <code>clbuildapp --check</code> with the <code>--force</code> argument   |
| <i>Stage 2: Volume Group Build</i>   |  |
| Volume group app01vg is not defined on node1 currently.<br>Currently this software will NOT create the volume group on the remote node automatically - you must do this first before attempting to build the application again.                | Create the volume group, logical volumes and file systems on the primary node ( <i>node1</i> )   |
| Volume group app01vgx is not defined on node2 currently.<br>Currently this software will NOT create the volume group on the remote node automatically - you must do this first before attempting to build the application again.               | Verify that the <b>vg.name</b> element in <b>appconf.xml</b> refers to an existing volume group, and correct it if necessary.<br>OR<br>Create the volume group on the secondary node ( <i>node2</i> ). |
| <i>Stage 3: Resource Allocation</i>  |  |
| The configuration file for this application has been altered but the <code>--vgbuild</code> stage has not yet been rerun. Do that first before running with the <code>--build</code> option.   | Run <code>clbuildapp --vgbuild</code> before running <code>clbuildapp --build</code>   |
| All logical volumes must be mounted for building! It appears that app01vg/lv01 is not mounted.   | The application volume group file systems must be mounted before running <code>clbuildapp --build</code>   |
| Specified application already defined on node2   | Rerun <code>clbuildapp --build</code> with <code>--force</code> argument<br>OR<br>Remove application using <code>clremoveapp</code> , then restart entire build process.                               |
| <i>Stage 4: Volume Group Synchronisation</i>   |  |
| The configuration file for app01 has been modified but the <code>clbuildapp</code> commands has not been rerun to validate the changes. You can ignore this warning and add a <code>--nochecksums</code> option to the command if you so wish. | Run <code>clbuildapp --build</code> before running <code>clbuildapp --sync</code> .  |
| TIME file for mysql on node2 exists - aborting Synchronisation   | Rerun <code>clbuildapp --sync</code> with <code>--force</code> argument.   |

## 6.8 Removing The Cluster Application

To remove or un-register application *appname*, execute:

```
# clremoveapp --application appname
```

The **clremoveapp** command only frees assigned Linuxha.net resources, it does not remove or alter file systems or volume groups associated with the application. The cluster must be running in order to execute **clremoveapp**.

## 6.9 Running The Cluster Application

If the value of the **global.autostart** element in **appconf.xml** is set to *yes*, then the application will be started when the cluster is formed. If the value is *no* or not specified or the cluster was formed with the **--noapps** argument then the application will have to be started manually.

The command to start a cluster application is **clrunapp**, it can be executed on either node. The cluster must be formed before executing **clrunapp**. To run application *appname*, execute:

```
# clrunapp --application appname
```

Table 14 describes some common **clrunapp** errors and their solutions:

Table 14: *clrunapp* Errors

| Error Message  | Solution  |
|--|---|
| Current application state is STARTED. An application can only be started if it is in state DOWN.   | Do nothing, application is already running  |
| The configuration file for app01 has been modified but the <b>clbuildapp</b> commands has not been rerun to validate the changes. You can ignore this warning and add a <b>--nochecksums</b> option to the command if you so wish. | Run the entire application rebuild process (use <b>--force</b> argument where necessary)  |
| Application app01 failed to start (RC=45)<br>Error:<br>Please run the following command manually for more information on failure:<br><code>/sbin/cluster/clstartapp --application apache --maxdelay 30 --verbose</code>            | <ul style="list-style-type: none"><li>• Execute <b>clhaltapp --force</b> to shut down the application.</li><li>• Set the value of <b>networks.network/ip</b> in <b>appconf.xml</b> to an unbound IP address.</li><li>• Run the entire application rebuild process (use <b>--force</b> argument where necessary)</li></ul> |
| Application app01 failed to start (RC=22)<br>Error:<br>Please run the following command manually for more information on failure:<br><code>/sbin/cluster/clstartapp --application apache --maxdelay 30 --verbose</code>            | The application file systems were mounted outside of DRBD, run <b>clhaltapp --force</b> to unmount the file systems, then rerun <b>clrunapp</b> .   |

## 6.10 Halting The Cluster Application

To halt application *appname*, execute:

```
# clhaltapp --application appname
```

If the application hangs while shutting down, use the **--force** argument

```
# clhaltapp --application appname --force
```

To force the application to halt if its application configuration was changed without the application being rebuilt, use the **--nochecksums** argument.

```
# clhaltapp --application appname --nochecksums
```

The **clhaltapp** command must be executed on the node on which the application is running.

## 7 Linuxha.net Cluster Setup Instructions

In this section, instructions for the set up of a Linuxha.net cluster will be provided as follows:

- Network card configuration
- Firewall setup
- SSH setup
- Cluster build
- Application build

The system to which the instructions in this section refer has the following initial configuration::

- Two machines with CentOS 4.2 installed.
- One hard disk in each machine.
- The portion of the disk reserved for cluster application installation is unpartitioned.
- Machine names are node1 and node2.
- Two network cards in each node, interface names are eth0 and eth1
- Each network card on a separate network
- IP address 192.100.0.1 bound to node1.eth0
- IP address 192.100.0.2 bound to node2.eth0

All commands discussed in this section are to be executed as the **root** user unless otherwise indicated.

### 7.1 Operating System Installation Notes

Specific operating system installation instructions are beyond the scope of this manual. However Linuxha.net requires that certain steps be taken during installation.

Table 15 describes these requirements.

Table 15: Linuxha.net Installation Requirements

| Install Category   | Requirements   |
|--------------------|--|
| Disk               | Linuxha.net requires at least one disk partition per application. <ul style="list-style-type: none"><li>• Use Disk Druid to partition disk.</li><li>• Allocate at least 2.5 GB for the root file system</li><li>• Do not partition the portion of the disk reserved for cluster applications.</li></ul>  |
| Network Interfaces | Linuxha.net requires at least two network cards per node. Each card has to be on a separate network. <ul style="list-style-type: none"><li>• Assign an address (192.100.0..1) to only one interface.</li><li>• Configure the other interface(s) as follows:<ul style="list-style-type: none"><li>• Activate on Boot - No</li><li>• Use DHCP - No</li></ul></li></ul> |
| Firewall           | Linuxha.net requires the SSH protocol. Apache requires the HTTP and HTTPS protocols. The firewall, if enabled, must allow these protocols  |
| SELinux            | Do not activate  |
| Applications       | Choose "Customise" option and install "Everything". Ensure that the applications to be clustered are installed.  |

## 7.2 Network Setup

This section will describe how to configure the two network cards in linuxha cluster node. As discussed in Section 4.3, each network card will be on a different network – a "public" network for regular network traffic, and a "private" network for data replication between cluster nodes.

The examples provided in this section will use the addresses shown in Table 16

Table 16:

|                 | node1     |             | node2     |             |
|-----------------|-----------|-------------|-----------|-------------|
|                 | Interface | IP Address  | Interface | IP Address  |
| Private Network | eth0      | 192.100.0.1 | eth0      | 192.100.0.2 |
| Public Network  | eth1      | 192.120.0.1 | eth1      | 192.120.0.2 |

There is no way to control which of the two network cards will be assigned to the *eth0* interface. Therefore it is necessary to verify that the correct card is physically connected to the private network.

1. Verify that an IP address has been assigned to *eth0* and no address assigned to *eth1* by executing the **ifconfig** command. This is to be done on both nodes. The output should be similar to the following, bolded lines show the IP address assignment status:

```
eth0      Link encap:Ethernet  HWaddr 00:10:DC:96:D1:71
          inet addr:192.100.0.1  Bcast:192.100.0.255  Mask:255.255.255.0
          inet6 addr: fe80::210:dcff:fe96:d171/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:9 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 b)  TX bytes:546 (546.0 b)
          Interrupt:11 Base address:0xcc00

eth1      Link encap:Ethernet  HWaddr 00:03:CE:88:8E:AB
          BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:9 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
          Interrupt:5 Base address:0xe800
```

2. Disconnect the network cable on *node1* from the private network switch, then execute the command **ethtool eth0**. The last line of output gives the state of the link, as shown by the bolded line in the example below:

```
Settings for eth0:
  Supported ports: [ TP MII ]
  Supported link modes:   10baseT/Half 10baseT/Full
                        100baseT/Half 100baseT/Full
  Supports auto-negotiation: Yes
  Advertised link modes:  10baseT/Half 10baseT/Full
                        100baseT/Half 100baseT/Full
  Advertised auto-negotiation: Yes
  Speed: 10Mb/s
  Duplex: Half
  Port: MII
  PHYAD: 1
  Transceiver: internal
  Auto-negotiation: on
  Supports Wake-on: pumbg
  Wake-on: d
  Current message level: 0x00000001 (1)
  Link detected: no
```

In this instance, there is no link detected which implies that the disconnected network card is interface *eth0*, which is as it should be. If the output shows **Link detected: yes**, then the other network card needs to be connected to the private network. It is recommended that a label be placed on the physical interface to facilitate future identification.

- Repeat steps 1 and 2 on *node2*.
- Assign an address to interface *eth1* on *node1* by editing */etc/sysconfig/network-scripts/ifcfg-eth1* as follows (changes are bolded):

```
DEVICE=eth1
BOOTPROTO=none
BROADCAST=192.120.0.255
NETMASK=255.255.255.0
NETWORK=192.120.0.0
IPADDR=192.120.0.1
HWADDR=00:03:CE:88:8E:AB
TYPE=Ethernet
ONBOOT=yes
```

- Repeat step 4 on *node2* using **IPADDR=192.120.0.2**.
- Edit the */etc/hosts* file on *node1* as follows (changes are bolded):

```
127.0.0.1          node1 localhost.localdomain localhost
192.100.0.2       node2
192.100.0.2       node2a
192.120.0.2       node2b
```

- Edit the */etc/hosts* file on *node2* as follows:(changes are bolded)

```
127.0.0.1          node1 localhost.localdomain localhost
192.100.0.1       node1
192.100.0.1       node1a
192.120.0.1       node1b
```

- Reboot both nodes and run *ifconfig* to verify IP address assignments.

### 7.3 Firewall Configuration

The firewall has to be configured to allow connection from the other cluster node.

- Edit the */etc/sysconfig/iptables* file on *node1* as follows (changes are bolded):

```
-A INPUT -j RH-Firewall-1-INPUT
-A FORWARD -j RH-Firewall-1-INPUT
-A RH-Firewall-1-INPUT -i lo -j ACCEPT
-A RH-Firewall-1-INPUT -s node2 -j ACCEPT
-A RH-Firewall-1-INPUT -s node2b -j ACCEPT
-A RH-Firewall-1-INPUT -p icmp --icmp-type any -j ACCEPT
-A RH-Firewall-1-INPUT -p 50 -j ACCEPT
-A RH-Firewall-1-INPUT -p 51 -j ACCEPT
```

- Repeat step 1 on the other node replacing *node2* with *node1* and *node2b* with *node1b*.

### 7.4 Disk Configuration

This section describes how to configure the disk to support the cluster applications. Each application will have one volume group consisting of a single physical volume.

- Invoke the disk partitioning utility by executing either:

```
# fdisk /dev/hda # IDE disk
```

or

```
# fdisk /dev/sda # SCSI disk
```

2. Enter **p** at the prompt to display the existing disk configuration. The output should be similar to that shown below.

| Device    | Boot | Start | End  | Blocks   | Id | System     |
|-----------|------|-------|------|----------|----|------------|
| /dev/hda1 | *    | 1     | 1275 | 10241406 | 83 | Linux      |
| /dev/hda2 |      | 1276  | 1341 | 530145   | 82 | Linux swap |

3. Create an extended partition occupying the rest of the disk by entering **n** to create a new partition and **e** to specify an extended partition. Hit enter twice to accept the default values for first and last cylinders.
4. Create a logical partition for the cluster application by entering **n** to create a new partition and **l** to specify a logical partition, hit enter to accept the default for the first cylinder, and enter the partition size in the format **+<size>M**, e.g. +1024M.
5. Change the type of the new partition to Linux LVM, by entering **t** to change the type, **5** as the partition number, then **8e** to specify Linux LVM.
6. Repeat steps 4 and 5 for each application, the partition number will increment by one (1) for each new logical partition. If a mistake is made, a logical partition can be deleted by entering **d** then the partition number.
7. Enter **p** when done to verify the disk layout. The output should be similar to that shown below:

| Device           | Boot | Start       | End         | Blocks           | Id        | System           |
|------------------|------|-------------|-------------|------------------|-----------|------------------|
| /dev/hda1        | *    | 1           | 1275        | 10241406         | 83        | Linux            |
| /dev/hda2        |      | 1276        | 1341        | 530145           | 82        | Linux swap       |
| /dev/hda4        |      | 1342        | 9964        | 69264247+        | 5         | Extended         |
| <b>/dev/hda5</b> |      | <b>1342</b> | <b>2646</b> | <b>10482381</b>  | <b>8e</b> | <b>Linux LVM</b> |
| <b>/dev/hda6</b> |      | <b>2647</b> | <b>3951</b> | <b>10482381</b>  | <b>8e</b> | <b>Linux LVM</b> |
| <b>/dev/hda7</b> |      | <b>3952</b> | <b>5256</b> | <b>10482381</b>  | <b>8e</b> | <b>Linux LVM</b> |
| <b>/dev/hda8</b> |      | <b>5257</b> | <b>6561</b> | <b>10482381</b>  | <b>8e</b> | <b>Linux LVM</b> |
| <b>/dev/hda9</b> |      | <b>6562</b> | <b>7807</b> | <b>10008463+</b> | <b>8e</b> | <b>Linux LVM</b> |

8. If the disk layout is correct, enter **w** to save the disk configuration and exit **fdisk**.
9. Initialise the logical partitions for use by the Logical Volume Manager with the following command:  

```
# pvcreate /dev/hda5 /dev/hda6 /dev/hda7 /dev/hda8 /dev/hda9
```

Use the partition names specific to your configuration.
10. Repeat steps 1 to 9 on *node2*.

## 7.5 Secure Shell (SSH) Setup

In order for data replication to take place between cluster nodes, they must be able to communicate via SSH without being prompted for a password. Instead of using passwords, SSH will authenticate using RSA which is based on public-key cryptography. In order for replication to take place between *node1* and *node2*, the **root** user must have associated with it a public/private key pair.

The procedure for setting up the key pairs for the **root** user on both servers is as follows:

1. On *node1*, enter the following command to define the public/private key pairs used by SSH.

```
# ssh-keygen -t rsa -b 1024
```

Hit enter in response to the following prompt to accept the default location for storing the keys:

```
Generating public/private rsa key pair.  
Enter file in which to save the key (/root/.ssh/id_rsa):
```

Hit enter in response to the following prompt so that no passphrase is created:

```
Created directory '/root/.ssh'.  
Enter passphrase (empty for no passphrase):
```

The public portion of the key pair will then be displayed.

2. Repeat step 1 on *node2*.
3. On *node1* add the loopback IP address (*localhost*) to the list of SSH known hosts (*~/.ssh/known\_hosts*) with the following command:

```
# ssh localhost ls
```

Enter **yes** in response to the following prompt:

```
The authenticity of host 'localhost (127.0.0.1)' can't be established.  
RSA key fingerprint is 53:86:82:ea:80:a5:cd:26:14:5e:46:07:b4:54:8c:10.  
Are you sure you want to continue connecting (yes/no)?
```

Enter the **root** password in response the following prompt:

```
Warning: Permanently added 'localhost' (RSA) to the list of known hosts.  
root@localhost's password:
```

4. Follow the same procedure to add the names and addresses of all interfaces local to *node1*, i.e. *node1*, *node1a*, *node1b*, *192.100.0.1*, *192.120.0.1*.
5. Follow the same procedure to add the names and addresses of interfaces on *node2* to the list of known hosts on *node1*, i.e. *node2*, *node2a*, *node2b*, *192.100.0.2*, *192.120.0.2*. Supply the **root** password for *node2* when prompted.
6. Repeat steps 3 and 4 on *node2* to add its loopback and local interface addresses, enter *node2's* **root** password when prompted. Repeat step 5, using *node1's* **root** password, to add *node1* addresses to the **known\_hosts** file on *node2*.
7. On *node1* copy the key pair list to the **authorized\_keys** file using the following command:

```
# cp ~/.ssh/id_rsa.pub ~/.ssh/authorized_keys
```

8. Open a terminal window session to *node2*, display *~/.ssh/id\_rsa.pub* and copy its contents – the *root@node2* public key.

**NOTE:** It should be possible to copy from the terminal window by highlighting the text with the mouse.

9. Open *~/.ssh/authorized\_keys* on *node1* in an editor, and append the *root@node2* public key.

**NOTE:** In the vi editor, type **o** to open a new line then click the right-mouse button to paste the key.

**IMPORTANT:** Make sure that the key is pasted as a single line (no line feeds).

10. Repeat steps 7 to 9 on *node2* to create *~/.ssh/authorized\_keys* and append the *root@node1* public key to it.
10. Enter the following command on *node1*:

```
# ssh node2 ls
```

If SSH is configured correctly, a listing of the root home directory on *node2* will be displayed without prompting for a password.

11. Verify SSH configuration on *node2*, by executing

```
# ssh node1 ls
```

## 7.6 Linuxha.net Software Requirements

Linuxha.net depends upon the following packages:

- gcc
- flex
- bison
- kernel.i686
- kernel-devel.i686

Use the **yum** command to install the packages on **both nodes** as follows:

```
# yum install flex bison gcc
# yum install kernel.i686 kernel-devel.i686
```

The second command installs an updated kernel, so the server must be rebooted before continuing.

## 7.7 Linuxha.net Installation

The linuxha.net RPM package can be downloaded from

[http://linuxha.net/wcodemgr2/index.cgi/show\\_project\\_downloads?linuxha](http://linuxha.net/wcodemgr2/index.cgi/show_project_downloads?linuxha)

The package has to be installed on **both nodes** of the cluster. The installation procedure is as follows:

1. Copy the downloaded package to the **/tmp** directory and execute the following:

```
# cd /tmp
# rpm -Uvh linuxha-1.0.4-1.noarch.rpm
```

The installation may take a minute or two to complete.

2. Verify the installation by executing the following to display the installation log:

```
# tail -12 ~/postinstall-linuxha-1.0.4.stdout
```

If the installation was successful, the following would be displayed:

```
DRBD Kernel module      : OK
Machine reset (dusp)    : OK
MII fallback binary     : OK
Send ARP fallback binary: OK
miitoollib              : OK
Crypt-CBC-2.08          : OK
Crypt-Blowfish-2.09    : OK
Net-ext-1.011           : OK
Net-Interface-0.04_2   : OK
```

3. The cluster script and manpages directories should be added to the PATH and MANPATH environment variables. To do so, edit the **~/.bash\_profile** file as follows (changes are bolded):

```
# User specific environment and startup programs
PATH=$PATH:$HOME/bin
PATH=$PATH:/sbin/cluster
MANPATH=/usr/local/cluster/man
export MANPATH
export PATH
unset USERNAME
```

4. Modify the MANSECT environment variable by editing the **/etc/man.config** file as follows (changes are bolded):

```
MANSECT          1:8:2:3:4:5:6:7:9:0p:1p:3p:tcl:n:l:p:o:1m
```

5. Log out and log back in.
6. Verify that the PATH variable is defined correctly by executing the following:

```
# which clstat
which should display:
/sbin/cluster/clstat
```

7. Verify that the MANPATH variable is defined correctly by executing the following:

```
# man clstat
```

8. Repeat steps 1 to 7 on *node2*.

## 7.8 Building The Cluster

Once Linuxha.net has been installed, the cluster can be built. The cluster is configured and resources are assigned to it during the build process. The cluster is built on the primary node (*node1*). The procedure only needs to be performed once and has two steps:

1. Configure the cluster
  - a) Create **clconf.xml** by copying **clconf.xml.example** using the following command:

```
# cp /etc/cluster/clconf.xml.example /etc/cluster/clconf.xml
```
  - b) Set the element values in **/etc/cluster/clconf.xml** on shown in Table 17, the numbers in brackets indicate to which node section the element belongs.

Table 17: *clbuild.xml* Element and Attribute Values

| Element / Attribute | Value                 |
|---------------------|-----------------------|
| global.name         | cluster1              |
| global.key          | topsecretkey          |
| global.drbd_network | a                     |
| node.name (1)       | node1                 |
| node.network (1)    | name="a" cards="eth0" |
| node.network (1)    | name="b" cards="eth1" |
| node.name (2)       | node2                 |
| node.network (2)    | name="a" cards="eth0" |
| node.network (2)    | name="b" cards="eth1" |

2. Build the cluster

Execute the following command:

```
# clbuild
```

See Sections 4.6 and 5.2 for additional information on the format of the **clconf.xml** file and the cluster build process.

## 7.9 Forming The Cluster

To start or form the cluster, execute the following on the **primary node (*node1*)**:

```
# clform
```

This command can be run as often as required, usually after the system is restarted.

See section 5.3 for additional information on forming a cluster.

## 7.10 Apache Cluster Application

The linuxha apache package, which automates some of the steps in building the Apache cluster application, can be downloaded from [????](#)

The cluster has to be built, but does not have to be running, prior to building the Apache cluster application.

Following are the instructions for building and running the Apache cluster application, **commands are to be executed on the primary node (node1) only, unless otherwise indicated**

### 1. Verify Apache installation

**This step is to be performed on both nodes.**

Execute the following verify that Apache is already installed:

```
# yum list httpd
```

If Apache is installed the output should contain a line similar to the following:

```
Installed Packages
httpd.i386                2.0.52-19.ent.centos4  installed
```

If Apache is not installed, execute the following:

```
# yum install httpd
```

### 2. Create Apache volume group.

**This command is to be executed on both nodes.**

Apache will be set up on a volume group named *apachevg*, it will have one physical volume - **/dev/hda5**.

Execute the following to create the *apachevg* volume group.

```
# vgcreate apachevg /dev/hda5
```

### 3. Install linuxha apache package

In the directory where the package was downloaded, execute

```
# rpm -U linuxha_apache-1.0.1-1.noarch.rpm
```

### 4. Create, mount and populate Apache logical volumes

The linuxha apache package installs scripts for setting up the Apache logical volumes in the **/etc/cluster/apache/build** directory.

Configure and run the build scripts:

a) Edit **/etc/cluster/apache/build/lvinfo** as follows:

```
DEF_VG=apachevg
DEF_PV=/dev/hda5

LV[0]="lv01:256:ext3:/apache"
```

b) Run the build script

```
# cd /etc/cluster/apache/build
# ./buildit -h node2
```

c. Copy the Apache icons to the Apache logical volume, by executing

```
# cp -rp /var/www/icons/* /apache/docs/icons
```

### 5. Define Apache configuration

To replicate Apache logs, edit **/apache/admin/conf/httpd.conf** as shown in the following extract (changes are bolded, comments omitted):

```
ErrorLog /apache/logs/error_log

LogLevel warn

LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined
```

```
LogFormat "%h %l %u %t \"%r\" %>s %b" common
LogFormat "%{Referer}i -> %U" referer
LogFormat "%{User-agent}i" agent
```

```
CustomLog /apache/logs/access_log common
```

## 6. Test Apache

Verify that the changes made to **httpd.conf** are valid by executing the Apache startup script as follows:

```
# /apache/admin/scripts/startapp
```

The following message will be displayed if apache was successfully started:

```
/apache/admin/scripts/apachectl start: httpd started
```

Shut down apache, by executing:

```
# /apache/admin/scripts/shutdown
```

## 7. Define apache cluster configuration

Set element values in **/etc/cluster/apache/appconf.xml** as shown in Table 18.

Table 18: Apache appconf.xml Elements and Attribute

| Element / Attribute   | Value                   |
|-----------------------|-------------------------|
| global.name           | apache                  |
| networks.network/name | b                       |
| networks.network/ip   | 192.120.0.10            |
| networks.checklist    | 192.120.0.2,192.100.0.2 |
| vg.name               | apachevg                |

Optional: To restrict the Apache daemon to listen only on the address defined by the **networks.network/ip** element change *Listen 80* to *Listen 192.120.0.10:80* in **httpd.conf**.

## 8. Build Apache cluster application

Execute the following on the primary node:

```
# cd
# clbuildapp --application apache --check
# clbuildapp --application apache --vgbuild
# clbuildapp --application apache --build
# clbuildapp --application apache --sync
```

See Section 5.2 for additional information, including troubleshooting suggestions, on the application build process.

## 9. Run Apache cluster application

If necessary, start the cluster using **clform** before running the Apache cluster application.

If the value of the **global.autostart** element in the Apache **appconf.xml** file is yes, then Apache will start automatically, otherwise execute the following:

```
# clrunapp --application apache
```

## 10. Test Apache cluster application

To verify that apache is running on the defined IP address use the **curl** command as follows:

```
# curl http://192.120.0.10
```

If Apache is running the following will be displayed:

```
<HTML>
<BODY>
<P>Hello World!</P>
</BODY>
</HTML>
```

## 7.11 MySQL Cluster Application

The cluster has to be built, but does not have to be running, prior to building the MySQL cluster application.

Following are the instructions for building and running the MySQL cluster application, **commands are to be executed on the primary node (*node1*) only, unless otherwise indicated**

### 1. Verify MySQL installation

Execute the following command on **both nodes** to verify that MySQL was installed.

```
# yum list mysql
```

If MySQL is installed the output should contain the following:

```
Installed Packages
mysql.i386                4.1.12-3.RHEL4.1        installed
```

If MySQL is not installed, execute the following on **both nodes**:

```
# yum install mysql
```

### 2. Disable automatic startup and shutdown.

The startup and shutdown of MySQL will be controlled by the cluster, so if the MySQL is configured to shut down and start up with the server, this feature must be disabled.

Delete the MySQL startup and shut down scripts by executing the following commands on **both nodes** (ignore any file not found errors):

```
# rm /etc/*d/S*mysql*
# rm /etc/*d/K*mysql*
```

You may be prompted to enter *y* to confirm the deletion of each file.

### 3. Define MySQL configuration

Modify **/etc/my.cnf** on **both nodes** as follows (changes are bolded):

```
[mysqld]
datadir=/mysql/data
socket=/var/lib/mysql/mysql.sock
# Default to using old password format for compatibility with mysql 3.x
# clients (those using the mysqlclient10 compatibility package).
old_passwords=1
port=3306      # Force the MySQL daemon to listen on a TCP socket

[mysql.server]
user=mysql
basedir=/var/lib

[mysqld_safe]
err-log=/var/log/mysqld.log
pid-file=/var/run/mysqld/mysqld.pid
```

### 4. Create MySQL volume group

MySQL will be set up on a volume group named *mysqlvg*, it will have one physical volume - **/dev/hda9**.

**Execute the following command on both nodes:**

```
# vgcreate mysqlvg /dev/hda9
```

### 5. Create logical volume

The MySQL volume group will have a single 1 GB logical volume named *data1v*.

Execute the following on the primary node (*node1*):

```
# lvcreate --name data1v --size 1G mysqlvg /dev/hda9
# mkfs -t ext3 /dev/mysqlvg/data1v
```

### 6. Mount and populate MySQL logical volume

- a) Execute the following commands on the primary node, starting MySQL initialises the data directory:

```
# /etc/init.d/mysqld stop      # Shut down mysql just in case
# mkdir /mysql
# mount -t ext3 /dev/mysqlvg/data1v /mysql/data
# mkdir /mysql/data
# chown mysql /mysql/data
# chgrp mysql /mysql/data
# chmod 755 /mysql/data
# /etc/init.d/mysqld start    # Initialise /mysql/data
# /etc/init.d/mysqld stop    # Shut down mysql
```

Note: If there is valid data in the MySQL databases then replace the last two commands with the following:

```
# cp -r /var/lib/mysql/* /mysql/data
# mv /var/lib/mysql /var/lib/mysql.moved
```

- b) Create the mount point on the secondary node by executing:

```
# mkdir /mysql
```

## 7. Configure MySQL cluster application

- a) Execute the following commands on the primary node (*node1*) to create the MySQL **appconf.xml** file.

```
# mkdir /etc/cluster/mysql
# cp /etc/cluster/appconf.xml.example /etc/cluster/mysql/appconf.xml
```

- b) Set element values in **appconf.xml** as shown in Table 19.

Table 19: MySQL *appconf.xml* Elements and Attributes

| Element / Attribute     | Value                    |
|-------------------------|--------------------------|
| global.name             | mysql                    |
| global.takeover         | normal                   |
| networks.network/net    | b                        |
| networks.network/ip     | 192.120.0.50             |
| vg.name                 | mysqlvg                  |
| application.startscript | /etc/init.d/mysqld start |
| application.stopscript  | /etc/init.d/mysqld stop  |

See Sections 4.6 and 6.2 for further information about the format of the **appconf.xml** file.

## 8. Configure MySQL Lems process:

- a) The behaviour of the MySQL Lems process is controlled by **/etc/cluster/mysql/lems.local.xml**. Create **lems.local.xml** by executing:

```
# cp /etc/cluster/lems.local.example /etc/cluster/mysql/lems.local.xml
```

- b) Edit **lems.local.xml** as shown in Table 20. The numbers in brackets indicate in which *check* section the element is located.

Table 20: MySQL *lems.local.xml* Elements and Attributes

| Element / Attribute | Value                                 |
|---------------------|---------------------------------------|
| globals/port        | 8809                                  |
| check.module (1)    | flag_check mysql                      |
| check.name (2)      | mysqld                                |
| check.module (2)    | procmon /etc/cluster/mysql/mysqld.xml |
| check.module (3)    | fsmon mysql                           |

- c) Create **/etc/cluster/apache/mysqld.xml** with the following content:

```
<?xml version="1.0"?>
<procmon>
  <global>
    <logdir>/var/log/cluster</logdir>
    <restarts>3</restarts>
    <resetwindow>3600</resetwindow>
    <restartcmd>/etc/init.d/mysqld restart</restartcmd>
  </global>

  <process>
    <label>MySQL Safe Database Server</label>
    <user>root</user>
    <process_string>/usr/bin/mysqld_safe</process_string>
    <min_count>1</min_count>
    <max_count>40</max_count>
  </process>
</procmon>
```

- d) Check that the Lems port is unique by executing:

```
# lems.pl --config /etc/cluster/mysql/lems.local.xml \
> --application mysql \ --verbose --check --file /dev/tty
```

If an error is displayed, change the value of the **globals/port** attribute in **lems.local.xml**.

- e) Copy the files to *node2* using the following commands:

```
# cd /etc/cluster/mysql
# scp lems.local.xml node2:$PWD
# scp mysqld.xml node2:$PWD
```

Additional information about **lems.local.xml** can be found in Section 6.2.

## 9. Build MySQL cluster application

Execute the following on the primary node:

```
# cd
# clbuildapp --application mysql --check
# clbuildapp --application mysql --vgbuild
# clbuildapp --application mysql --build
# clbuildapp --application mysql --sync
```

See Section 6.7 for additional information, including troubleshooting suggestions, on the application build process.

## 10. Run MySQL cluster application

If necessary, start the cluster using **clform**, before running the MySQL cluster application.

If the value of the **global.autostart** element in the MySQL **apponf.xml** file is **yes**, then MySQL will start automatically, otherwise execute the following:

```
# clrunapp --application mysql
```

## 11. Test MySQL cluster application

Verify that MySQL is running on its defined IP address by connecting to that address with the **mysql** program. However, first a connection must be made via the localhost address to grant permission to connect using the defined IP address.

- a) Grant permission to connect to MySQL (**mysql>** is the **mysql** application prompt):

```
# mysql --user=root
mysql> use mysql;
mysql> grant all on *.* to 'root'@'192.120.0.50';
```

- b) Create and populate a test table, then exit the MySQL client:

```
mysql> use test;
mysql> create table t1 (f1 int, f2 varchar(15));
mysql> insert into t1 values(1, 'first row');
```

```
mysql> exit;
```

- c) Connect to the MySQL cluster application over its defined IP address, by executing:

```
# mysql --host=192.120.0.50 --port=3306 --user=root
```

- d) If the `mysql>` prompt is displayed then the connection was successful, as a further test, display the contents of the `t1` table by executing:

```
mysql> use test;  
mysql> select * from t1;
```

- e) If the table contents are displayed then all is well, type `exit` to close the MySQL client.

## 7.12 Samba Cluster Application

The `linuxha samba` package, which automates some of the steps in building the Samba cluster application, can be downloaded from [????](#)

The cluster has to be built, but does not have to be running, prior to building the Samba cluster application.

Following are the instructions for building and running the Samba cluster application, **commands are to be executed on the primary node (*node1*) only, unless otherwise indicated.**

### 1. Verify Samba installation

Execute the following on **both nodes** to verify that Samba is installed:

```
# yum list samba
```

If Samba is installed, the output should include the following:

```
Installed Packages  
samba.i386                3.0.10-1.4E.2            installed
```

If Samba is not installed, execute the following on **both nodes**:

```
# yum install samba
```

### 2. Configure firewall to allow Samba traffic

**This step is to be carried out on both nodes.**

Add the following lines to `/etc/sysconfig/iptables`:

```
-A RH-Firewall-1-INPUT -p tcp --dport microsoft-ds -j ACCEPT  
-A RH-Firewall-1-INPUT -p tcp --dport netbios-ssn -j ACCEPT
```

Execute the following command to restart the firewall:

```
# /etc/init.d/iptables/restart
```

### 3. Create Samba volume group

Samba is to be set up on a volume group named `sambavg`, which consists of one physical volume - `/dev/hda6`.

Execute the following on **both nodes** to create the `sambavg` volume group:

```
# vgcreate sambavg /dev/hda6
```

### 4. Install linuxha samba package

Download the `linuxha samba` package to `node1` and execute the following to install it:

```
# rpm --install linuxha_samba-1.0.1-1.noarch.rpm
```

### 5. Create, mount and populate samba logical volumes.

The `linuxha samba` package installs scripts for setting up the samba logical volumes in the `/etc/cluster/samba/build` directory. To manage the logical volumes, the build scripts have to be configured and executed.

- a) Edit `/etc/cluster/samba/build/lvinfo` as shown below:

```
DEF_VG=sambavg
```

```

DEF_PV=/dev/hda6
DEF_PE=8

LV[0]="cfg:128:ext3:/samba/cfg"
LV[1]="logs:128:ext3:/samba/logs"
LV[2]="shares:5000:ext3:/samba/shares"

```

b) Run the build script

```

# cd /etc/cluster/samba/build
# ./buildit -h node2

```

## 6. Configure Samba

Edit the `/samba/cfg/smb.conf` file as shown in the following extract (changes are bolded, note the semicolons):

```

[global]
;
; Identity
;
; Allow several Samba servers on different subnet without conflicts
; socket address = 192.120.0.110
; interfaces = 192.120.0.110/255.255.255.0
bind interfaces only = no

```

## 7. Test Samba

Execute the following:

```

# /samba/cfg/scripts/start
# ps -ef|grep smbd|grep -v grep

```

If Samba is running, a result similar to the following will be returned:

```

root  6742      1  0  01:15 ?    00:00:00 /usr/sbin/smbd -s /samba/cfg/smb.conf -D
root  6743  6742  0  01:15 ?    00:00:00 /usr/sbin/smbd -s /samba/cfg/smb.conf -D

```

Stop Samba by executing:

```

# /samba/cfg/scripts/stop

```

## 8. Define Samba cluster configuration

Edit `/etc/cluster/samba/appconf.xml` and set the XML elements to the values shown in Table 21.

Table 21: Samba appconf.xml Elements and Attributes

| Element /Attribute    | Value                   |
|-----------------------|-------------------------|
| global.name           | samba                   |
| networks.network/name | b                       |
| networks.network/ip   | 192.120.0.110           |
| networks.checklist    | 192.120.0.2,192.100.0.2 |
| vg.name               | sambavg                 |

Optional: To restrict the samba daemon to listen only on the address defined by the `networks.network/ip` element uncomment the **socket address** and **interfaces** lines in `smb.conf`.

Additional information about the format of `appconf.xml` can be found in Sections 4.6 and 6.2.

## 9. Build Samba cluster application

Execute the following on the primary node:

```

# cd
# clbuildapp --application samba --check
# clbuildapp --application samba --vgbuild
# clbuildapp --application samba --build

```

```
# clbuildapp --application samba --sync
```

See Section 6.7 for additional information, including troubleshooting suggestions, on the application build process.

#### 10. Run Samba cluster application

In necessary, start the cluster using `clform` before running the samba cluster application.

If the value of the `global.autostart` element in the Samba `appconf.xml` file is `yes`, then Samba will start automatically, otherwise execute the following:

```
# clrunapp --application samba
```

#### 11. Test Samba cluster application

Verify that the Samba cluster application is working properly by connecting to a Samba share using `smbclient`, as follows:

##### a) Create Unix user

```
# useradd -g users sambal
```

##### b) Grant user permission to use Samba (passwords are not echoed)

```
# smbpasswd -c /samba/cfg/smb.conf -a sambal
New SMB password: qwerty1
Retype new SMB password: qwerty1
```

##### c) Create test file

```
# echo "this is a test file" > /samba/shares/tmp/test1.txt
```

##### d) Connect to Samba share and display contents of test file

```
# smbclient //192.120.0.110/tmp -U sambal
Password: qwerty1
```

If the connection was successful, results similar to the following will be displayed:

```
Domain=[LINUXHAPC] OS=[Unix] Server=[Samba 3.0.10-1.4E.2]
```

##### e) Display the contents of the test file by executing:

```
smb: \> more test1.txt
```

If Samba can display the file contents, it should return something similar to the following:

```
getting file \test1.txt of size 20 as /tmp/smbmore.sSCWeI (4.9 kb/s)
this is a test file
```

##### f) Close smbclient by entering:

```
smb: \> exit
```

## 7.13 OpenLDAP Cluster Application

The cluster has to be built, but does not have to be running, prior to building the OpenLDAP cluster application.

Following are the instructions for building and running the OpenLDAP cluster application, **commands are to be executed on the primary node (node1) only, unless otherwise indicated.**

#### 1. Verify OpenLDAP installation

Execute the following command on **both nodes** to verify that OpenLDAP was installed.

```
# yum list openldap openldap-clients openldap-servers
```

If OpenLDAP is installed the output should contain the following:

```
Installed Packages
openldap.i386                2.2.13-4                installed
openldap-clients.i386       2.2.13-4                installed
openldap-servers.i386       2.2.13-4                installed
```

If one or more of the OpenLDAP packages is not installed, execute the following on **both nodes**:

```
# yum install openldap openldap-clients openldap-servers
```

## 2. Disable automatic startup and shutdown

The startup and shutdown of OpenLDAP will be controlled by the cluster. If OpenLDAP is configured to startup or shutdown with the server, this feature must be disabled.

Execute the following on **both nodes** to delete the OpenLDAP automatic startup and shutdown scripts:

```
# rm /etc/rc*.d/K*ldap*
# rm /etc/rc*.d/S*ldap*
```

## 3. Configure firewall to allow OpenLDAP traffic

**This step is to be performed on both nodes**

Add the following lines to the **/etc/sysconfig/iptables** file:

```
-A RH-Firewall-1-INPUT -p tcp --dport ldap -j ACCEPT
-A RH-Firewall-1-INPUT -p udp --dport ldap -j ACCEPT
-A RH-Firewall-1-INPUT -p tcp --dport ldaps -j ACCEPT
-A RH-Firewall-1-INPUT -p udp --dport ldaps -j ACCEPT
```

Restart the firewall by executing:

```
# /etc/init.d/iptables restart
```

## 4. Configure OpenLDAP

**This step is to be performed on both nodes**

a) Add the following lines to **/etc/openldap/ldap.conf**:

```
URI ldap://node1 ldap://192.120.0.55
BASE dc=linuxha,dc=net
TLS_CACERTDIR /openldap/cacerts
```

b) Edit the **/etc/openldap/slapd.conf** file as follows (extract shown in context, changes are bolded):

```
#####
# ldbm and/or bdb database definitions
#####

database          bdb
suffix            "dc=linuxha,dc=net"
rootdn            "cn=Manager,dc=linuxha,dc=net"
# Cleartext passwords, especially for the rootdn, should
# be avoided. See slapasswd(8) and slapd.conf(5) for details.
# Use of strong authentication encouraged.
rootpw            secret
directory         /openldap/data
```

## 5. Create OpenLDAP volume group

The OpenLDAP volume group will be named *ldapvg* and contain a single partition - **/dev/hda8**. Execute the following command on **both nodes** to create *ldapvg*:

```
# vgcreate ldapvg /dev/hda8
```

## 6. Create and populate logical volume

The OpenLDAP volume group will have two logical volumes - *cacerts* and *data*; which will be mounted on **/openldap/cacerts** and **/openldap/data** respectively.

Execute the following commands to create and mount the logical volumes on the primary node:

```
# lvcreate --name cacerts --size 256M ldapvg /dev/hda8
# lvcreate --name data --size 1G ldapvg /dev/hda8
# mkdir /openldap /openldap/cacerts /openldap/data
# mkfs -t ext3 /dev/ldapvg/cacerts
# mkfs -t ext3 /dev/ldapvg/data
```

Create the mount point on the **secondary node (node2)** by executing:

```
# mkdir /openldap /openldap/cacerts /openldap/data
```

## 7. Populate OpenLDAP logical volumes

Start and then shut down OpenLDAP to initialize files in **/openldap/data** by executing the following command:

```
# slapd -f /etc/openldap/slapd.conf
# killall slapd
```

## 8. Define OpenLDAP cluster configuration

Execute the following commands to create the OpenLDAP **appconf.xml** file.

```
# mkdir /etc/cluster/openldap
# cp /etc/cluster/appconf.xml.example /etc/cluster/openldap/appconf.xml
```

Set element values in **appconf.xml** as shown in Table 22

Table 22: OpenLDAP appconf.xml Elements and Attributes

| Element / Attribute     | Value                                       |
|-------------------------|---|
| global.name             | openldap                                    |
| networks.network/name   | b   |
| networks.network/ip     | 192.120.0.55                                |
| networks.checklist      | 192.120.0.2,192.100.0.2                     |
| vg.name                 | ldapvg                                      |
| application.startscript | /usr/sbin/slapd -f /etc/openldap/slapd.conf |
| application.stopscript  | /usr/bin/killall slapd                      |

Optional: Remove the reference to <http://node1> in the **URI** entry of the **/etc/openldap/ldap.conf** file so that the cluster application IP address (192.120.0.55) is the only LDAP server.

## 9. Configure OpenLDAP Lems process

- a) The OpenLDAP Lems process is configured by **/etc/cluster/openldap/lems.local.xml**. Create **lems.local.xml** by executing:

```
# cp /etc/cluster/lems.local.example /etc/cluster/openldap/lems.local.xml
```

- b) Edit **lems.local.xml** as shown in Table 23. The numbers in brackets indicate in which check section the element is located.

Table 23: OpenLDAP lems.local.xml Elements and Attributes

| Element / Attribute | Value                                   |
|---------------------|---|
| globals.port        | 8802                                    |
| check.module (1)    | flag_check openldap                     |
| check.name (2)      | openldap                                |
| check.module (2)    | procmon /etc/cluster/openldap/slapd.xml |
| check.module (3)    | fsmon openldap                          |

- c) Create **/etc/cluster/openldap/openldap.xml** with the following content:

```
<?xml version="1.0"?>
<procmon>
  <global>
    <logdir>/var/log/cluster</logdir>
    <restarts>3</restarts>
    <resetwindow>3600</resetwindow>
    <restartcmd>
      /usr/bin/killall slapd; sleep 3; /usr/sbin/slapd -f /etc/openldap/slapd.conf
    </restartcmd>
  </global>

  <process>
    <label>OpenLDAP Server</label>
    <user>root</user>
    <process_string>/usr/sbin/slapd -f </process_string>
```

```
<min_count>1</min_count>
<max_count>10</max_count>
</process>
</procmon>
```

d) Verify Lems port by executing:

```
# lems.pl --config /etc/cluster/slaped/lems.local.xml \
> --application slapd \ --verbose --check --file /dev/tty
```

If an error is displayed, change the value of the **globals/port** attribute in **lems.local.xml**.

## 10. Build OpenLDAP cluster application

Execute the following on the primary node:

```
# cd
# clbuildapp --application openldap --check
# clbuildapp --application openldap --vgbuild
# clbuildapp --application openldap --build
# clbuildapp --application openldap --sync
```

See Section 6.7 for additional information, including troubleshooting suggestions, on the application build process.

## 11. Run OpenLDAP cluster application

If necessary, start the cluster using **clform**, before running the OpenLDAP cluster application.

**OpenLDAP and Fedora-DS cannot be running simultaneously**, if necessary shut down Fedora-DS before starting OpenLDAP by executing the following:

```
# clhaltapp --application nsslapd
```

If the value of the **global.autostart** element in the OpenLDAP **appconf.xml** file is yes, then OpenLDAP will start automatically, otherwise execute the following:

```
# clrunapp --application openldap
```

## 12. Test OpenLDAP

Execute the following to verify that OpenLDAP is running on its specified IP address:

```
# ldapsearch -H ldap://192.120.0.155 -x -b '' \
> -s base '(objectclass=*)' namingContexts
```

If OpenLDAP is running, the following will be displayed:

```
# extended LDIF
#
# LDAPv3
# base <> with scope base
# filter: (objectclass=*)
# requesting: namingContexts
#
#
dn:
namingContexts: dc=linuxha,dc=net
```

## 7.14 Fedora Directory Server (Fedora-DS)

The cluster has to be built, but does not have to be running, prior to building the Fedora-DS cluster application.

Following are the instructions for building and running the Fedora-DS cluster application, **commands are to be executed on the primary node (node1) only, unless otherwise indicated**.

### 1. Verify Fedora-DS installation

Check whether Fedora-DS is installed by executing the following on **both nodes**:

```
# yum list fedora-ds
```

If installed, the command output should include the following:

```
Installed Packages
fedora-ds.i386                1.0.1-1.RHEL4                installed
```

If Fedora-DS is not installed then follow this procedure to install it on **both nodes**:

- a) Download the Fedora-DS package to the **/tmp** directory. The package is available from <http://directory.fedora.redhat.com/download/fedora-ds-1.0.1-1.RHEL4.i386.opt.rpm>.
- b) Execute the following to install:

```
# cd /tmp
# rpm -Uvh fedora-ds-1.0.1-1.RHEL4.i386.opt.rpm
```

## 2. Setup Fedora-DS by executing:

```
# cd /opt/fedora-ds/setup
# ./setup
```

Detailed setup instructions are available from <http://directory.fedora.redhat.com/wiki/Setup>.

For the purposes of this configuration choose Typical install mode and accept the default responses to the prompts except:

- Directory server identifier: *userdetails*
- Suffix: *dc=linuxha,dc=net*

Stop Fedora-DS by executing:

```
# /opt/fedora-ds/slaped-userdetails/stop-slaped
```

**Important:** The fully qualified host name specified during setup must be the default - *localhost.localdomain* and not the real host name.

## 3. Create Fedora-DS volume group

The Fedora-DS volume group will be named *nsslapdvg* and contain the **/dev/hda7** partition.

Execute the following on **both nodes**:

```
# vgcreate nsslapdvg /dev/hda7
```

## 4. Create, mount and populate Fedora-DS logical volume

- a) Create a 1 GB logical volume named *data* by executing:

```
# lvcreate --name data --size 1G nsslapdvg /dev/hda7
# mkfs -t ext3 /dev/nsslapdvg/data
```

- b) Copy contents of **/opt/fedora-ds/slaped-userdetails** to the new file system by executing:

```
# mount -t ext3 /dev/nsslapdvg/data /mnt
# cd /opt/fedora-ds/slaped-userdetails
# find . | cpio -pumd /tmpmnt
```

- c) Rename **slaped-userdetails** directory, **on both nodes**, by executing:

```
# cd
# mv /opt/fedora-ds/slaped-userdetails \
> /opt/fedora-ds/slaped-userdetails.moved
# mkdir /opt/fedora-ds/user-details
# chmod 755 /opt/fedora-ds/user-details
```

- d) Mount new file system (on primary node only) by executing:

```
# umount /mnt
# mount -t ext3 /dev/nsslapdvg /opt/fedora-ds/slaped-userdetails
```

## 5. Define Fedora-DS cluster configuration

Execute the following commands to create the Fedora-DS **appconf.xml** file.

```
# mkdir /etc/cluster/nsslapd
# cp /etc/cluster/appconf.xml.example /etc/cluster/nsslapd/appconf.xml
```

Set element values in **appconf.xml** as shown in Table 24.

*Table 24: Fedora-DS appconf.xml Elements and Attributes*

| Element / Attribute     | Value  |
|-------------------------|--|
| global.name             | nsslapd                                      |
| networks.network/name   | b  |
| networks.network/ip     | 192.120.0.144                                |
| networks.checklist      | 192.120.0.2,192.100.0.2                      |
| vg.name                 | nsslapdvg                                    |
| application.startscript | /opt/fedora-ds/slapd-userdetails/start-slapd |
| application.stopscript  | /opt/fedora-ds/slapd-userdetails/stop-slapd  |

## 6. Configure Fedora-DS Lems process

- a) The configuration file for the Fedora-DS Lems process is **/etc/cluster/nsslapd/lems.local.xml**. Create **lems.local.xml** by executing:

```
# cp /etc/cluster/lems.local.example /etc/cluster/nsslapd/lems.local.xml
```

- b) Edit **lems.local.xml** as shown in Table 25. The numbers in brackets indicate in which **check** section the element is located.

*Table 25: Fedora-DS lems.local.xml Elements and Attributes*

| Element / Attribute | Value                                    |
|---------------------|--|
| globals/port        | 8806                                     |
| check.module (1)    | flag_check nsslapd                       |
| check.name (2)      | nsslapd                                  |
| check.module (2)    | procmon /etc/cluster/nsslapd/nsslapd.xml |
| check.module (3)    | fsmon openldap                           |

More information about the format of **lems.local.xml** can be found in Section 6.2.

- b) Check that the Lems port is unique by executing:

```
# lems.pl --config /etc/cluster/nsslapd/lems.local.xml \
> --application nsslapd \ --verbose --check --file /dev/tty
```

If an error is displayed, edit the value of the **globals/port** element in **lems.local.xml**.

- c) Create **/etc/cluster/nsslapd/nsslapd.xml** with the following content:

```
<?xml version="1.0"?>
<procmon>
  <global>
    <logdir>/var/log/cluster</logdir>
    <restarts>3</restarts>
    <resetwindow>3600</resetwindow>
    <restartcmd>
      /opt/fedora-ds/slapd-userdetails/restart-slapd
    </restartcmd>
  </global>

  <process>
    <label>Directory Server</label>
    <user>nobody</user>
    <process_string>ns-slapd.*userdetails</process_string>
    <min_count>1</min_count>
    <max_count>40</max_count>
  </process>
</procmon>
```

## 7. Build Fedora-DS cluster application

Execute the following on the primary node:

```
# cd
# clbuildapp --application nsslapd --check
# clbuildapp --application nsslapd --vgbuild
# clbuildapp --application nsslapd --build
# clbuildapp --application nsslapd --sync
```

See section 6.7 for additional information, including troubleshooting suggestions, on the application build process.

## 7. Run Fedora-DS cluster application

If necessary, start the cluster using `clform` before running the Fedora-DS cluster application.

**Fedora-DS and OpenLDAP cannot be running simultaneously**, if necessary shut down OpenLDAP before starting Fedora-DS by executing the following:

```
# clhaltapp --application openldap
```

If the value of the `global.autostart` element in the Fedora-DS `appconf.xml` file is `yes`, then Fedora-DS will start automatically, otherwise execute the following:

```
# clrunapp --application nsslapd
```

## 8. Test Fedora-DS cluster application

Execute the following to verify that Fedora-DS is running on its specified IP address:

```
# ldapsearch -H ldap://192.120.0.144 -x -b '' \
> -s base '(objectclass=*)' namingContexts
```

If Fedora-DS is running, results similar to the following will be displayed:

```
# extended LDIF
#
# LDAPv3
# base <> with scope base
# filter: (objectclass=*)
# requesting: namingContexts
#
#
dn:
namingContexts: dc=linuxha,dc=net
namingContexts: o=NetscapeRoot

# search result
search: 2
result: 0 Success

# numResponses: 2
# numEntries: 1
```