A Practical Guide to XEN High Availability

Configuring Enterprise Virtualization on SUSE Linux Enterprise Server

Sander van Vugt

March 2010

Published by Books4Brains, March 2010

P.O. Box 345
3830 AJ Leusden
The Netherlands
www.books4brains.com

ISBN 9789072389084

Disclaimer
The author and publisher of this book have made every effort to make this book as complete and accurate as possible. The author and publisher make no representations or warranties of any kind as to the accuracy or the contents of this book and accept no liability whatsoever with respect to any loss or damages that may arise from the information contained in this book or from the use of any programs described in this book.

The author and publisher do not guarantee that the information in this book will continue to work with future versions of the describedsoftware.

Trademarks
SUSE is a trademark of Novell, Inc. All other product names printed in this book are trademarks of their respective companies.

Table of Contents

Introduction

Foreword

Most books get printed and once you've got the printed copy, you'll have to make do with it. The reason is that the publisher will print enough books for the coming two years or so, which means that a year and a half after the publication date of the book, you'll be stuck with an outdated book. This book is different.

This book is a print-on-demand book for which only a minimal stock exists. The reason behind that, is that the subject matter that is covered in this book, is of an extremely fast changing subject, high availability of Xen virtual machines. I don't want you to work on an outdated book within a couple of months. I want you to have access to the most recent version of the book. How will this work? Easy. I'll keep on working on the book, which means that the content gets updated continuously. The July 2010 version of this book, will be different from the March 2010 version you are currently reading. If you need the most recent version, just ask for it.

How do you ask for the most recent version? Easy. If you've purchased this book, visit www.books4brains.com/xenha and register the book. I will maintain a database with the mail addresses of everyone that purchased this book, and the date you've registered your purchase. For a full year after your purchase, you will have access to updates, free of charge. There is a catch though. I'm a lazy person, so you have to ask for it. (You will receive contact details after registering.) Just ask, and I will send you your personal copy of the book. The book will be watermarked, which hopefully discourages you to share the PDF with others. In my opinion that's fair enough. Believe me, this book was not written for the money. If you divide the invested time by the expected revenue, I would have been better off taking an evening job at the local McDonald's restaurant in my home town. So if someone use the content, I'd like the benefit of compensation for the long hours of work I've put in this book.

This concept is new, so I hope you like it. If it succeeds, I want to apply the concept to all my future books as well. If it doesn't work, I'll try something else to keep you updated with the most up-to-date information.

Enjoy the book!

Sincerely,

Sander van Vugt
www.sandervanvugt.nl

About the author

Sander van Vugt is an independent technical trainer and consultant, specialising in Linux storage, virtualization, HA clustering and performance optimization. As such, he has been invited to do work all around the globe. Sander is also an author who has written over 40 books. He's a regular contributor to different magazines and websites. You can reach Sander at this mail address mail@sandervanvugt.nl, or his website www.sandervanvugt.nl.

About the reader

This book is written for anyone who wants to create an affordable and stable solution for high availability of Xen virtual machines. To get the most out of this book, the reader should have a good working knowledge of Linux. The book uses SUSE Linux Enterprise as the example distribution. Although the configuration is applicable to other distributions as well, there are no guarantees on this.

Acknowledgements

This book has been made possible with the aid of the following people:

- Mike de Laat of IKW, the Xen HA Implementation at his site has inspired me to document what I've done, which was the initial start of this book.
- Rob Bastiaansen of Books4Brains, Rob is always open for a good idea, even if from a commercial perspective it might not be viable.
- Erno de Korte of Open Horizons. The Open Horizons Community has a book fund that helps authors who want to publish about a certain subject realizing the book without having to make a major investment first.
- Corné Groesbeek of NetCB. Corné spent his weekend making transcribing my horrible mix between Dutch and English (Dunglish) into decent English.
- The people on the Pacemaker mailing list. Thanks for answering all my stupid questions and thanks for working on this great software.

1 Understanding the Xen HA solution

In this chapter you'll get an introduction to Xen virtualization and more specifically, the Xen HA solution. Firstly, you'll get to learn the benefits of working with Xen as your virtualization solution. I'll compare Xen to other virtualization solutions, such as VMware and KVM. Next, you'll learn what exactly is meant by Xen High Availability. In the last part of this chapter, you'll read what hardware to purchase to build a decent Xen High Availability Solution.

Note: this book is about creating a Xen High Availability solution. The solution described will however work for other environments as well. It is likely that in the future, KVM will become increasingly important as a virtualization solution. Using the setup described in this book, you can create High Availability for KVM virtual machines as well. I'll probably dedicate a chapter to that subject in one of the next versions of the book.

1.1 Free Virtualization with High Availability and optimal Performance

Once the decision has been made to implement a virtualised solution, the majority of industry players choose VMware. There are several reasons for doing this, of which the fear of implementing a solution that cannot be managed effectively is an important one. Too many customers pay for the perception of security and hence buy VMware. In this book you'll read about an alternative, which is as good as VMware but much cheaper: Xen on SUSE Linux Enterprise Server 11, combined with Pacemaker High Availability.

SLES or something else?

In this book I will be discussing the configuration on SUSE Linux Enterprise Server. You're by no means bound to use the same configuration or purchase the solution offered by SUSE Linux Enterprise Server. You can use the components discussed here on free software, such as OpenSUSE or Ubuntu as well. If you don't want to use SUSE Linux Enterprise Server, I'd recommend using OpenSUSE instead. The structure of this distribution is quite similar to the software discussed here, whereas there will be some differences if you're using other distributions like Ubuntu.

Let's be honest. The Open Source virtualization solution that we're going to build on top of SUSE Linux Enterprise Server 11, is not as easy and accessible as the same solution based on VMware. If you're looking for a solution where anyone can migrate a virtual machine easily from one host to another, using an intuitive graphical interface, the SUSE-Xen solution isn't there yet. You need to know command line Linux in order to work with this solution, and from time to time you

will need to work with hard to use commands at the Linux console, or with cryptical XML files.
Does this mean that you shouldn't use it? Not at all! A solution based on VMware also needs a specialist to be installed correctly. Often VMware solutions fail because a skilled system administrator was mislead by the easy to use interface and has messed things up. If you start any virtualization project, you need to know what you are doing!

1.2 Why Xen?

Before starting a virtualization project that is based on Xen, you have to ask why you want to do it with Xen. Xen has a reputation of being an inferior and less reliable solution than VMware.

It's true that the market share for Xen is not as big as that of VMware, but Xen is more than just another geek open source project that has no or little support. Let's have a look at the history of Xen, and the companies currently supporting it's development.

Xen was originally developed at the university of Cambridge in the United Kingdom. The scientists responsible for the development of the Xen hypervisor (which is the core component of Xen virtualization), included it in their XenServer, which they took to market with their company XenSource. Various other vendors were involved in the open source Xen project. Amongst these, chip developers such as AMD and Intel, vendors from the Linux market, such as Red Hat and Novell, and even Microsoft has been involved with Xen. Their current HyperV virtualization solution is based a large part on Xen! So if you thought Xen wasn't serious enough, that's not true. Xen enjoys large industry support , which has increased wht Citrix having acquired XenSource.

Since the development of the initial versions of the Xen hypervisor, lots of changes took place on the virtualization market. As mentioned, Citrix acquired XenSource. Does that mean Xen is owned by Citrix? By no means! Xen is still an open source project, of which major players of the open source market is involved in ! Citrix is an important player in the development of XenServer, but not the only one. Many important Linux vendors - such as Novell, Red Hat and Oracle - offer a Xen virtualization stack that is fully supported. Xen being open source offers many benefits, including all vendors being able to use the open source solutions that have been created by any one of the other vendors. For example, if Citrix - or any other vendor - makes a change to the Xen hypervisor, anyone can implement and use it in his own solution.

There are some differences between Citrix XenServer and the Novell Xen solution that we discuss in this book. The differences are mainly related to management utilities. With Citrix, you get XenCenter, an easy to use management interface that makes working in a virtual environment very easy. Apart from that, Citrix offers

Citrix Essentials, an extensive - but paid for - solution that allows you to perform all management tasks on your virtualization environment. In the SUSE environment, currently, there's only the Virtual Machine Manager, known to not be the most sophisticated tool for managing virtual machines.
If you want to take advantage of the benefits of Xen, but at the same time, you require an easy to use management solution, you might be better of choosing Citrix Xen. If you're not afraid of the Linux command line and you're willing to work with tools that are not as sophisticated, you can save a lot of money using Novell Xen.

Apart from Xen, there is another major player on the Linux virtualization market, KVM, the Kernel Virtual Machine. The big difference between Xen and KVM, is KVM's integration into the Linux kernel itself. To be able to virtualize with KVM, you need only one kernel module. To virtualize with Xen, you need to load a Xen hypervisor which in turn loads the Linux kernel. Thus Xen is a lot more complicated than KVM. The development of KVM however is currently not as far as that of Xen, therefor it is recommended to build your solution on Xen. Even Red Hat, the company that owns KVM technology, still offers Xen as their default virtualization solution.

1.3 Xen Technical benefits

Up to now, we haven't talked about the technical benefits of Xen. These are mainly in the way that Xen handles virtualization. The makers of Xen introduced a revolutionary new approach to virtualization, which is known as paravirtualization. In paravirtualization, a software interface is used for virtual machines. This software interface is used to communicate directly with the underlying hardware. By using this interface, the approach of emulation can be avoided. Emulation should be avoided, since all hardware instructions generated by virtual machines have to be trapped and translated. This process has a very high price in performance terms. In short: paravirtualization makes your virtualization solution a lot faster. Currently, Xen is the only player in the virtualization space that has completely implemented paravirtualization. Other solutions are still trying to get to this point.

There is however something you need to know about paravirtualization. To use paravirtualization, you need a special kernel in the operating system that you want to install as a virtual machine on. This kernel is not available for all operating system. There is a paravirtualization kernel for Novell NetWare and Linux, but you won't find one for Windows. You can still virtualize Windows in Xen, but you'll have to run it in full virtual mode. By using this mode, the virtualized operating system has to go through a layer of emulated drivers, which will cost additional performance. Since Xen is using virtualization support on the CPU in this mode, the performance of Windows virtual machines is still quite good. Novell has also made available the Virtualization Driver Pack, a set of drivers that can be installed in Windows for I/O devices, which means that at least on your network and your

storage adapters you can use paravirtualization technology. These drivers are only available as an additional purchase. For more information on the Novell Virtual Machine Driver Pack, you can refer to:
http://www.novell.com/products/vmdriverpack.

1.4 Understanding the Xen High Availability Solution

Before we start talking about hardware and configurations, let's try to understand what exactly it is that we are going to build. To keep it simple, imagine one virtual machine. You can run this one virtual machine on a single host. However, if the host goes down, the virtual machine won't run anymore. This problem will only get bigger if several virtual machines are involved.

To avoid a situation like this, it is recommended to build a high availability solution, using at least two physical machines. In such a solution, you'll configure a cluster that gives you the high availability. The cluster will be installed on the hosts in the cluster and the task of the cluster is to monitor the other nodes. Should one of the hosts in the cluster go down, the cluster will know it. Even better: if a virtual machine goes down, the cluster will also know it. The advantage is that the cluster will ensure that the virtual machines that were running on the host that just went down, will be restarted on an alternative node as soon as possible.
So through the configuration of a high availability solution for Xen environments, you can ensure minimum downtime. This does not mean however that you'll have uninterrupted services. It will take some time for the cluster to notice that a virtual machine or host went down, and to start up the virtual machine on the alternative node. With the current state of the available software, there is no way that you can avoid this. This is likely to change in future releases of Xen. (In fact, the recently released Xen 4.0 hypervisor can do this. You will see this feature in the major Xen-based solutions very soon.)

1.4.1 Required Hardware

Now that you know the benefits of using a virtualization solution that is built on Xen, let's have a look at the hardware required build such a solution. To begin with, you need a central storage area. The centralized storage should be accesable by all nodes in the cluster. In most environments, this central storage area is implemented as a SAN. You may hear people talk about other solutions such as a NAS or a NFS Server, however for the solution described in this book, only a SAN is going to work.

Implementing a SAN does not have to cost a lot, nowadays some very affordable SAN solutions are available, such as Dell's MD3000i SAN which has been tested for this environment and works well. Alternatively, you could build your own open source SAN on iSCSI. To install a Linux Server with storage attached, configure iSCSI to share this storage and you have a cheap iSCSI SAN. There's no need for an expensive Fibre Channel solution, iSCSI works just as well in most

environments. The Appendix at the end of this book describes how to create such an open source SAN solution.

Installing the SAN is not enough. After installion, you need to ensure that the SAN is connected correctly to the virtualization hosts. A single connection is not enough, because if the cable for this connection breaks, everything will stop.
You need redundancy. It is recommended that redundancy is implemented on the following components:

- Disk controllers in the SAN
- Network cards in the SAN
- Switches on the network that connects to the SAN
- Network cards or iSCSI HBA's in the hosts that connect to the SAN.

Apart from the SAN and the hardware infrastructure, you also need a couple of host machines that you can run the virtual machines on. For a high availability environment, you need at least two machines, it is recommended that you use a minimum of three machines. Should you only have two machines, and one fails, you run the risk of the remaining node being overloaded when all your virtual machines are being loaded on the single remaining node. To decrease this risk it is therefor recommended to use more than two hosts. If, however, you have budgetry constraints, the solution can work on a two node cluster.

1.4.2 Choosing the Appropriate Hardware for the Host Machines

One of the most important hardware components you have to configure on the host, is RAM. It's not that difficult to calculate the required RAM. Add the RAM requirements of all the virtual machines together, round it up to the next reasonable configuration of RAM and you're good. For example: If you want to install four Windows virtual machines that require 4 GB each, and three OES servers that need 2 GB each, as well as 8 Linux servers that need 2 GB, your total RAM requirement for all the virtual machines would be 34 GB. Since you are going to spread this workload over 3 host machines, you would need a theoretical minimum of 16 GB per host. However, configuring 16 GB per host will cause a problem if one node fails, since you'll have only 32 GB available on the two remaining hosts. It is there for recommended to use at least 24GB per node in this scenario.

In other words: in your calculations make sure that the amount of RAM you need to run all of your virtual machines, is offered by the total amount of host computers minus 1. If you need a total of 80GB for your virtual machines, make sure that you have three hosts that have 48 GB of RAM each. With this configuration you ensure that the cluster remains stable if , after the failure of a host, all the virtual machines are migrated across the remaining two hosts!

Once you have calculated the RAM requirement, you'll have to decide what to do with the network cards in your server. It is likely that your server has multiple network ports, so if you're using iSCSI over a normal network card, you need two iSCSI interfaces to start with. Next, you'll need a network that interfaces the user as well and it's a good idea to configure that in redundancy. In Linux terminology, you will be implementing NIC bonding.

In a Xen HA environment, there are two approaches to NIC bonding. The first approach would be to bundle the network cards so that they make one huge trunk with double or fourfold bandwidth. This however doesn't work well in a Linux environment. It is recommended to put two network ports together in a redundant pair. If you have four network cards in your server, you would create two of these in redundant pairs. This will also gives you another benefit: you can manually load balance between servers. If you have a very busy web server, and other servers are not that busy, you can assign one bond to the web server and the other bond to be shared by all the other servers. Right now however, you need to know what hardware to buy for your servers. Just make sure that your server have at least two, but preferably four network interfaces available for the LAN.

Last, but not least, you have to think about CPU power. The ideal scenario would be if you could assign one CPU core per virtual machine. If you have two quad-core CPU's per server, you would have 24 cores.

The rest of the hardware configuration in your server is not as important. It would be a good idea to have a reasonable minimum of local storage, so that you can put an image file or ISO on the storage area. The bulk of the storage however will be on the SAN. Having an optical drive is optional as well. If you don't have an optical driver, you can create an installation server which works just as well.

2 Configuring the Hosts

At this point, I'll assume that you have all the hardware in place and you are ready to start working on the software part of the installation. Configuring the software is a multi-step procedure:

- Installation of the host servers
- Creating the LUN's on the SAN
- Connecting the host servers to the SAN, including multipath configuration
- Configuration of networking on the hosts, including NIC bonding and bridge configuration.
- Installation of the cluster
- Creating Xen virtual machines
- Connecting the Xen virtual machines to the cluster

In this chapter we will discuss what you need to do to configure the host machines, including the configuration of networking on the hosts.

2.1 Installation of the host machines

To start, you need to install some software on the host servers. In this book, we'll use SUSE Linux Enterprise Server (SLES) 11 for this purpose. Assuming that you have some basic Linux knowledge already, I won't explain step-by-step what you need to do to get SLES 11 installed. In some aspects of the installation, you might be required to make specific selections.

SLES 11 will prompt you early in the configuration phase what exactly you want to install. You can choose between a full installation, the installation of a paravirtual machine, or installation as a Xen host. You might think that you need the latter option, but that's not the case. Since the Xen host installation pattern installs the minimum number of components only, its better to use the Full installation option and add the Xen packages to your installation. By selecting the full installation option, you sure that some basic components are configured automatically and that makes the overall configuration a lot easier. You could add any of the components at a later stage. Another benefit of selecting the full installation option, is that it will give you a graphical user interface. A Linux expert wouldn't need a GUI to configure a server, but I've written this book for novices as well and they might want to work with the GUI tools. It is simply easier to run the tools locally rather than tunneling through SSH.

While installing the servers, you don't need to do anything in particular with regard to networking. We'll take care of the network configuration later in this chapter, so just accept the default settings which will configure one network card which will get it's IP address from a DHCP server. You'll change this later.

After installing the servers, you need to take care of the SAN. As every SAN is different, I can't be very specific on which configuration steps exactly you need to perform. The only thing that should be clear is that you need a LUN as the storage back-end for every virtual machine. So if you want 50 virtual machines, you will need 50 LUN's. It is possible to have multiple machines share the same LUN, but the configuration will be less flexible, so it is not recommended. You should check that your SAN supports the creation of so many LUN's. Some SAN vendors restricts the amount of LUN's that you can create on the SAN. You might have to buy additional SAN licensing to be able to add the required amount of LUN's. It's not necessary to create all LUN's now, you can create some LUN's later. Ensure sure that you have two or three LUN's at this moment, which will allow you to configure the first couple of virtual machines.

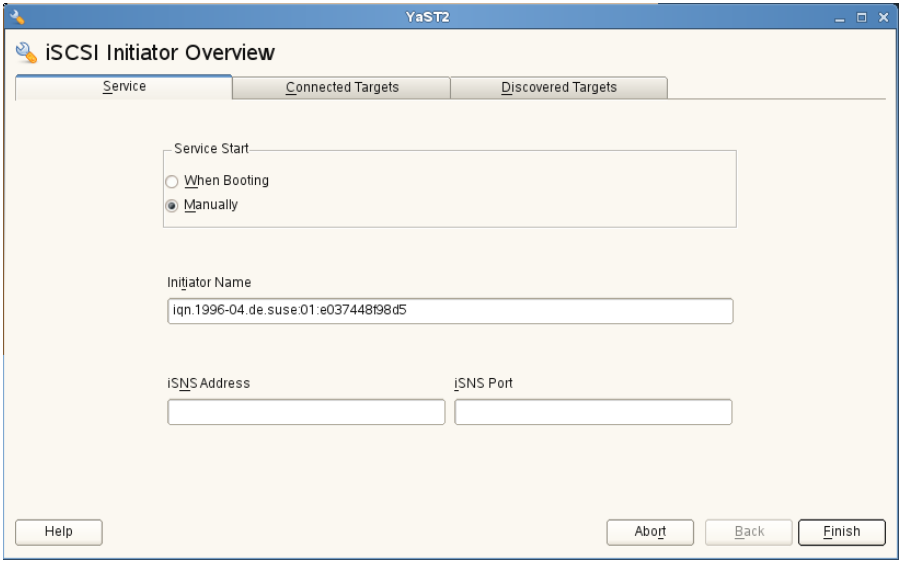
2.2 Connecting the Servers to the SAN

After installing the host operating system on all of the servers, you can connect the servers to the SAN. We'll assume that your servers are equipped with two network card each to connect to the iSCSI SAN. Although the individual steps can be slightly different in some environment, the procedure described here, gives a good indication of what is required. You'll need this procedure for environments where you want to connect to the SAN using normal Ethernet network cards, or specific iSCSI HBA's. If you're using a Fibre Channel, the configuration will be different, as the Fibre Channel HBA connects directly to the SAN, without requiring additional network configuration.

2.2.1 Creating the iSCSI Connection

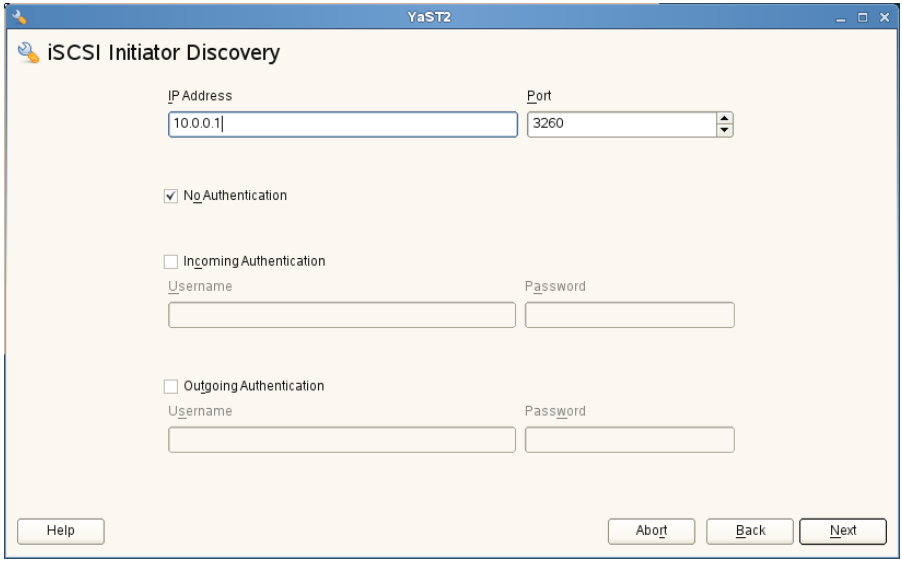
If you're using iSCSI, you can configure SUSE to make the iSCSI connection. Let there be no confusion about this, you'll configure iSCSI from SUSE, not from the HBA (which may offer you some configuration options from the BIOS also). The HBA configuration is required only if you intend to boot the host operating systems from the SAN, which is specifically not the case in a Xen HA environment.
The procedure below, describes how to make the iSCSI connection from a SUSE host, assuming that you've already set up IP on your computer.

1. From YaST, select Network Services > iSCSI Initiator. YaST will probably tell you that iSCSI hasn't been installed yet and ask if you want to install it now. It is recommended that you select the installation at this point.
2. After installation, you'll see the iSCSI configuration window. On this window, indicate that you want to start iSCSI automatically when booting the server. (As everything else depends on the successful creation of this connection, you definitely want it to start automatically when booting).
3. You'll now see the iSCSI configuration window. From here, select the option that starts iSCSI automatically when booting the server.



To begin with, you need to indicate that the iSCSI service is to be started automatically on all nodes, so change the default option that starts it manually.

4. Navigate to the Discovered Targets tab and click Discovery. In this interface, enter the first IP address that you need to connect to the SAN. If your SAN has two (or more) IP addresses , enter the second as well as any other IP addresses. If you get a warning against a second connection to the same SAN, you can safely ignore that.



Now enter all the IP addresses you're using to connect to the SAN.

5. Activate the Connected Targets tab. On this tab, you can see the current iSCSI connections, which should have the status “On Boot”. That means that the driver is loaded when booting, but nothing else will happen automatically. To make sure that the iSCSI connection is really created, you have to select each connection and click Toggle Startup until the startup mode has changed to Automatic. This will ensure that the iSCSI connection is really initiated when booting, making all the iSCSI disk devices visible after booting your computer. In the following listing, you can see how to make the disk devices visible using the lsscsi command; the iSCSI disks are listed with the VIRTUAL-DISK disk type.

Listing: The lsscsi command shows you the iSCSI disk as the virtual disk type.

```
node2:/ # lsscsi
[0:0:0:0]  disk   VMware,  VMware Virtual S 1.0   /dev/sda
[2:0:0:0]  cd/dvd NECVMWar VMware IDE CDR10 1.00  /dev/sr0
[3:0:0:0]  disk   IET      VIRTUAL-DISK     0     /dev/sdb
[3:0:0:1   disk   IET      VIRTUAL-DISK     0     /dev/sdc
[4:0:0:0]  disk   IET      VIRTUAL-DISK     0     /dev/sdd
[4:0:0:1]  disk   IET      VIRTUAL-DISK     0     /dev/sde
```

2.2.2 Configuring Multipath

After configuring the network cards to connect to the SAN, you'll notice that all of the LUN's that you have created on the SAN are offered twice. If you have access to the SAN through two different controllers, you'll see each LUN listed four times! For each of the connections that your server makes, you'll get a corresponding disk device on the host computer. So your LUN0 on the SAN, might be known as /dev/sdb, /dev/sdc, /dev/sdd as well as /dev/sde on your host computer! You cannot use these devices as they are too specific. Each of the drives depends on a specific connection and if the connection fails, you'll loose the corresponding drive as well. So you need a solution that is more generic and doesn't depend on a specific connection. This solution is offered by using multipath.
Before configuring multipath, it's important to decide which multipath to use. In most environments, you can use the open source Linux multipath driver, as well as the multipath driver that is provided by your SAN vendor. In general it's a better idea to use the Linux multipath driver. The reason? Linux multipath is stable and offers you all you need, and it's maintained by your Linux distribution as well. That means that the driver is patched each time your Linux server is updated. When using the multipath driver of your vendor, you run the risk of needing to recompile it after every major upgrade of the kernel. That's something you really don't want to do in a production environment.

To enable the multipath driver on SUSE Linux Enterprise Server, perform the following steps:

1. It is very likely that the multipath driver has already been installed, but it is recommended that you confirm this first. Hence, use the zypper search multipath command. In the listing below you can see the result of this command. In this listing, it's the first line of the result that matters, the i at the start of the line tells you that the multipath software has been installed. If zypper doesn't show you the i, use zypper install multipath to install it now.

   Listing: Use zypper search multipath to find out if multipath has already been installed

```
node1:/ # zypper search multipath
Loading repository data...
Reading installed packages...
S | Name           | Summary                                                    | Type
--+----------------+------------------------------------------------------------+----------
i | multipath-tools | Tools to Manage Multipathed Devices with th-> | package
  | multipath-tools | Tools to Manage Multipathed Devices with th-> | srcpackage
```

2. After verifying that multipath has been installed, you need to make sure that it starts automatically from the various runlevels. You can do this by using the insserv multipathd command. You don't need boot.multipath, it is only needed if you want to boot from the SAN. Using insserv ensures that multipath is started when your computer reboots. It doesn't start it at this moment though. To start multipath manually now, use the /etc/init.d/multipath start command.

3. Multipath should now be running. You can check this with the multipath -l command. You can see the result of this command in the listing below.

In the output of the multipath -l command, you can see that the connections have been established successfully. The first lines show you the names of the devices that have been created. For each device you will se two names. First, there is the long hexadecimal number, which corresponds to the names that you can find in the directory /dev/disk/by-id. These names give you a unique method to communicate to disk devices. After the by-id name, you can also see the device mapper names (dm-1). Device mapper is the layer between multipath and the kernel, a layer that is also responsible for the LVM (Logical Volume Manager), which is covered later in this book. You can use either of these names to refer to the LUN on the SAN. These names will work in all circumstances, no matter which distribution you are using.

Listing: The multipath -l command shows if you have created the multipath connections succesfully.

```
node2:/ # multipath -l
149455400000000000000000001000000950500000f000000 dm-1 IET,VIRTUAL-DISK
[size=10.0G][features=0][hwhandler=0][rw]
\_ round-robin 0 [prio=-1][active]
 \_ 3:0:0:1 sdc 8:32  [active][undef]
\_ round-robin 0 [prio=-1][enabled]
 \_ 4:0:0:1 sde 8:64  [active][undef]
149455400000000000000000001000000 8e0500000f000000 dm-0 IET,VIRTUAL-DISK
[size=10G][features=0][hwhandler=0][rw]
\_ round-robin 0 [prio=-1][enabled]
 \_ 3:0:0:0 sdb 8:16  [active][undef]
\_ round-robin 0 [prio=-1][enabled]
 \_ 4:0:0:0 sdd 8:48  [active][undef]
```

If you're lucky, you will see a nice and decent list like in the preceding listing. This however isn't always the case. Should you not receive a listing like this, the first thing you should do is update your system. I've seen problems on SLES 11, especially after a fresh installation. Often this is resolved once the system has been updated. In some situations, you may need a specific configuration for your SAN. Such a configuration can be created, containing very specific parameters on protocols to be used in the configuration file /etc/multipathd.conf.

Below you can see the configuration that I've used on several occasions to connect to a Dell MD3000i SAN. Notice however that this configuration probably won't

work on other SAN's as they make use of different configurations. In some instances you might not need a specific configuration for your SAN. Many storage solutions work out of the box, after just updating the SLES 11 operating system.

Listing: To enter specific parameters that your SAN needs to use, create /etc/multipathd.conf

```
devices {
        device {
                vendor                  "DELL"
                product                 "MD3000i"
                path_grouping_policy    "group_by_prio"
                getuid_callout          "/lib/udev/scsi_id -g -u -d /dev/%n"
                prio_callout            "/sbin/mpath_prio_rdac /dev/%n
                hardware_handler "1 rdac"
                path_checker            rdac
                prio                    rdac
                path_selector           "round-robin 0"
                failback        immediate
                rr_weight               priorities
                no_path_retry           queue
                rr_min_io               100
                product_blacklistLUN_Z
                }
}
```

The settings you use in the configuration file, are dependant on the way you connect to the SAN. If it doesn't work with the default settings, it might be a good idea to look for an example file for your specific device. If based on that, you still can't get it to work, have a look at the man page of multipathd.conf for information about specific parameters. Especially consider the path_checker variable, this parameter determines which kernel module is loaded for a specific SAN and not every module is usable for every SAN environment.

2.3 Configuring the LAN Interfaces

Now it's time to have a look at the LAN. In order to create a Xen configuration, you'll need to create a bridge device on your host operating system and there are several ways to do that.

The bridge device is used as a virtual switch, which is contacted by each of the virtual machines, including the host operating system. Since the bridge device is an essential component of your servers configuration, it's recommended to configure fault tolerance on this device. There are two approaches to do this, on the host or on the virtual machines. In either solution, you'll need a bond device, which is a software device (referred to as the bond device) to which at least two network cards are connected. You'll connect at least two network cards, but more than two could be required. An advantage of this, is better redundancy. If one of the devices configured in the bond fails, the other device just takes over.

From the viewpoint of network bonding, you can go in two directions. One solution is to join the network cards in the interface and have them share the workload on your server. This method however doesn't work well in some configurations. The second approach is to configure the bond device in an active/passive configuration. This method is supported and offers you the best possible redundancy.

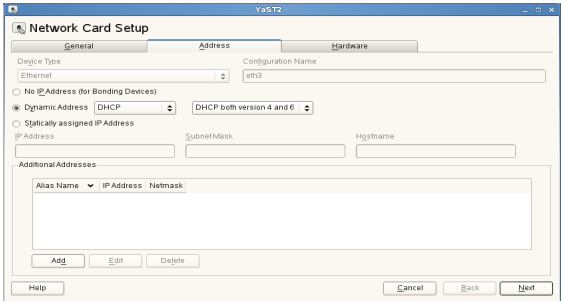
You can also go for a completely different approach, in which the bond device is not created on the host operating system, but in the virtual machines themselves. In some network environments, this just seems to work better. If you decide to configure a bond in the virtual machines, you can skip the next section which is about configuring the bond device on the host machine. Instead, just skip ahead to section 2.5 where you learn how to create a network bridge and configure bonding later, after you have installed the virtual machines.

In the following subsections you'll learn how to configure the bond device on either the host operating system, or on the virtual machines.

When configuring bonding on the host environment, you'll need to create the bond device on the host and connect the physical network cards that connect you to the LAN to that device. After doing this, you'll need to connect the bond device to the virtual bridge. For a large part, you can perform this procedure from YaST. You'll have to configure some of the elements by editing the configuration files directly though.

As a first step, you need to create a configuration for the network cards that you want to use in the bond device. From YaST, this is easy to do: YaST offers an option to configure a network card to be used in a bond device. This procedure is described in the following steps:

1. Start YaST, select Network Devices and next choose Network Settings. Then select the first network card you want to use and click Edit to modify its settings. Have a look at the figure below for the YaST screen from which to do this.



From YaST, it's easy to configure a network card to be used in a bond device.

2. Now select the option No IP Address (Bond Devices) and click Next to save the configuration. You are now back in the main window, where you can see that the configuration for the network card is marked as NONE.

3. Repeat this procedure for all network cards that you want to include in the bond device. As active/passive really is the only way that is supported well, it doesn't make much sense to include more than two network cards per bond device.

Now you need to check the configuration files for your network cards. You can find these files in the directory /etc/sysconfig/network, and you can recognize them by their name, which looks like ifcfg-ethX. Below you can see what the file looks like after configuring it for use in a bond device.

Listing: Example configuration file for the eth interfaces

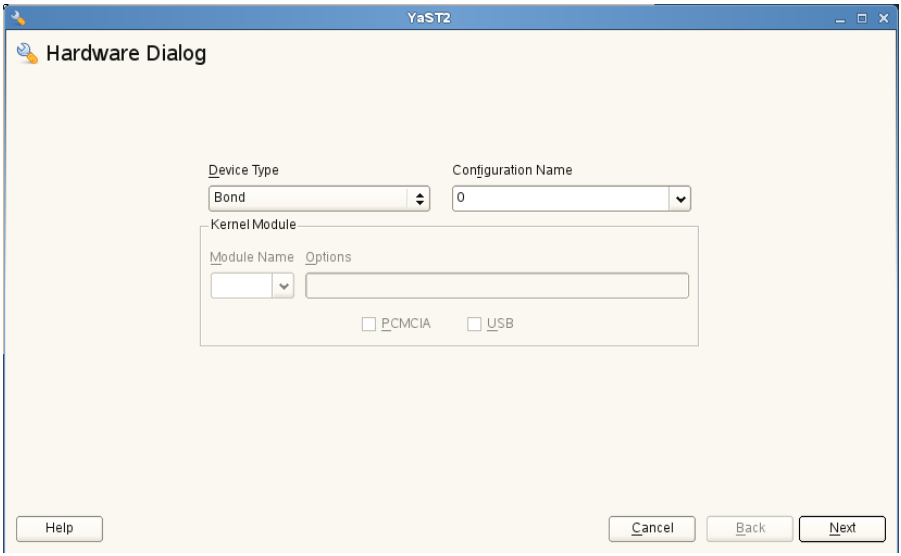
```
BOOTPROTO='none'
BROADCAST=''
ETHTOOL_OPTIONS=''
IPADDR=''
MTU=''
NAME='82575GB Gigabit Network Connection'
NETMASK=''
NETWORK=''
REMOTE_IPADDR=''
STARTMODE='off'
USERCONTROL='no'
```

2.4 Creating a Bond Device

If you want to configure bonding at the host level, it's time to do it now. You can do this from the Network Settings > Overview window. To create the bond device, you need to perform a few steps. To start with, you'll have to create the device itself. Next, you need to specify which network cards you want to connect to the device. Finally, you'll specify which protocol is to be used by the bond device. You can find a detailed description of this process below.

1. Start YaST and select Network Devices > Network Settings > Overview. Then click Add to create a new interface.

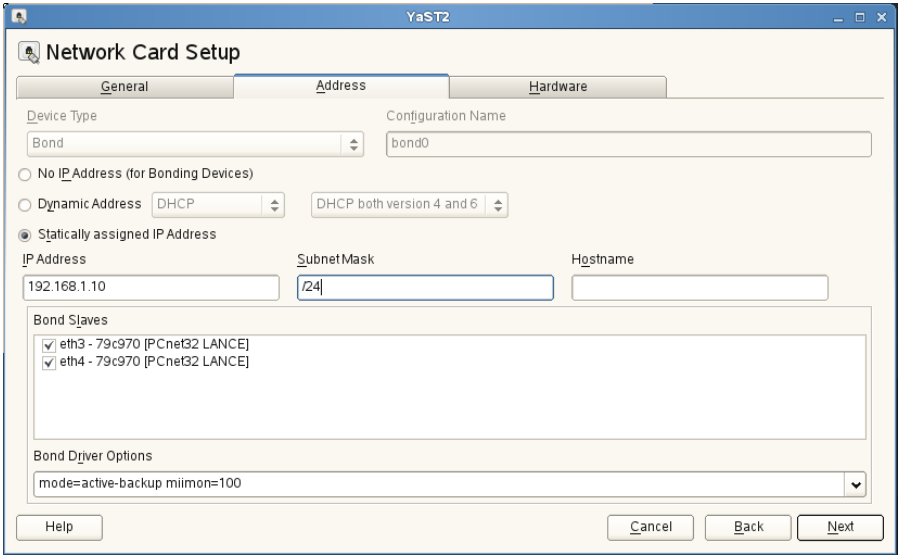
2. From the drop-down list under Device Type, select the Bond device type. For the configuration name, you can accept the default choice, 0. This will create the device with the name bond0. Tools as ifconfig, or ip address will show the device with this name.



From the drop-down list, select the device type bond to create a bond device.

3. Now you have to add a configuration to the device. First, you select the network cards that are going to be used by the device. You can choose any of the network cards that don't have their own IP configurations. Then you need to select which bonding protocol can be used. If this is your first try with NIC bonding, it's recommended not to change the default protocol. Once you've verified that the default protocol is working, you can test if another protocol will work in your environment (this may require additional configuration of

your switch though). There's something else you have to be aware of: YaST doesn't like a bond device without an IP address, so you need to add a random temporary IP address (which you will remove again in the next step of this procedure).



Now select the network cards to be used in the bond device, as well as the bonding protocol you want to use.

4. At this point you can finalize the procedure in YaST, to make sure that the configuration is written to disk. As a result, the corresponding configuration file will be created. You'll find it as /etc/sysconfig/network/bond0, and its content will look as follows:

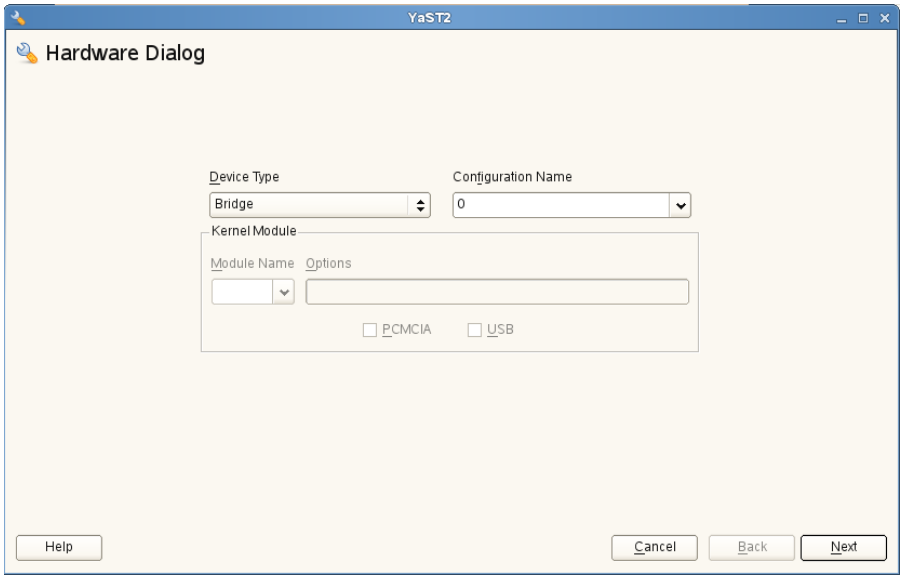
   Listing: Example configuration of the bond0 interface

```
BONDING_MASTER='yes'
BONDING_MODULE_OPTS='mode=active-backup'
BONDING_SLAVE0='eth0'
BONDING_SLAVE1='eth1'
BOOTPROTO='none'
BROADCAST=''
ETHTOOL_OPTIONS=''
IPADDR=''
MTU=''
NAME=''
NETMASK=''
NETWORK=''
REMOTE_IPADDR=''
STARTMODE='auto'
USERCONTROL='no'
```

5. As you can see in the listing, the configuration file that has been created for the bond device, still contains an IP address and a netmask. You need to remote these from the configuration file, using your favourite editor (not YaST!). After removing the IP address, use ifdown bond0, followed by ifup bond0 to bring it up again. You can now continue the procedure by creating the network bridge.

2.5 Creating the Network Bridge

It is now time for the last step to be performed. You need to configure the network bridge. You can also accomplish this task from YaST: select Network Settings > Overview and click Add. This time, you have to select the Device Type Bridge from the drop-down list. Just leave the configuration name 0. The figure below shows what this looks like from YaST.



You can also create the network bridge using YaST

Here you'll see an interface that looks a lot like the interface when you created the bond device. In this interface, you have to specify which network cards you want to connect to the bridge. That would be the bond0 device that you've just created if you want to set up bonding on the host computers. In case you don't want to do that, select one of the Ethernet cards that are offered by YaST and configure them in the bridge. In this section I'll assume that you want to configure a bridge on top of a bond device.

Now assign an IP address and subnet mask to the bridge and click Next to write the configuration to your system. This will create the configuration file, which has the name /etc/sysconfig/network/ifcfg-br0. The default configuration file is not good

enough though, you have to add a few parameters. The listing below shows what the configuration file should look like, make sure that your configuration is similar.

Listing: Example Configuration for the Bridge Interface

```
BOOTPROTO='static'
BRIDGE='yes'
BRIDGE_FORWARDDELAY='0'
BRIDGE_PORTS='bond0'
BRIDGE_STP='off'
BROADCAST=''
ETHTOOL_OPTIONS=''
IPADDR='192.168.1.101/24'
MTU=''
NAME=''
NETWORK=''
REMOTE_IPADDR=''
STARTMODE='auto'
USERCONTROL='no'
PREFIXLEN='24'
DEVICE='switch'
TYPE='Bridge'
BRIDGE_HELLOTIME='2'
BRIDGE_MAXAGE='20'
BRIDGE_PATHCOST='19'
```

Now that you have changed the configuration file, you can restart the bridge. To do this, use the command ifdown br0, followed by ifup br0. This will make your network configuration operational. If external nodes are available, you should be able to ping them now. You can now reach the ifconfig command to check the network configuration. The result of this command should look as follows:

Listing: Your network configuration now should look as follows

```
node1:/ # ifconfig
bond0     Link encap:Ethernet  HWaddr 00:0C:29:5C:88:70
          UP BROADCAST RUNNING MASTER MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)

br0       Link encap:Ethernet  HWaddr 00:0C:29:5C:88:70
          inet addr:192.168.1.10  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)

eth1      Link encap:Ethernet  HWaddr 00:0C:29:5C:88:5C
          inet addr:10.0.0.10  Bcast:10.0.0.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MUaddr 00:0C:29:5C:88:66
          inet addr:10.0.0.11  Bcast:10.0.0.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:6 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:360 (360.0 b)  TX bytes:0 (0.0 b)
          Interrupt:19 Base address:0x20a4
```

```
eth3      Link encap:Ethernet  HWaddr 00:0C:29:5C:88:70
          UP BROADCAST RUNNING SLAVE MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 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:16 Base address:0x2424

eth4      Link encap:Ethernet  HWaddr 00:0C:29:5C:88:70
          UP BROADCAST RUNNING SLAVE MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 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:17 Base address:0x24a4

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:4 errors:0 dropped:0 overruns:0 frame:0
          TX packets:4 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:200 (200.0 b)  TX bytes:200 (200.0 b)
```

Check if the bridge is functional and able to communicate with external nodes. If this is the case, you can configure the remaining nodes. If the bridge has communication problems, you have to fix them before continuing. Make sure that you've resolved all issues. The configuration has to work smoothly before you continue.

***Note**: An error that occurs often, is that the bond0 device is configured from YaST with the option "No IP Address (Bond device)". This option doesn't only configure your network card without an IP address, but also makes sure that it's not started automatically, which means that it's unusable. To check if this is the case, you should read the configuration file /etc/sysconfig/network/ifcfg-bond0 the parameter startmode should have the value "auto"*

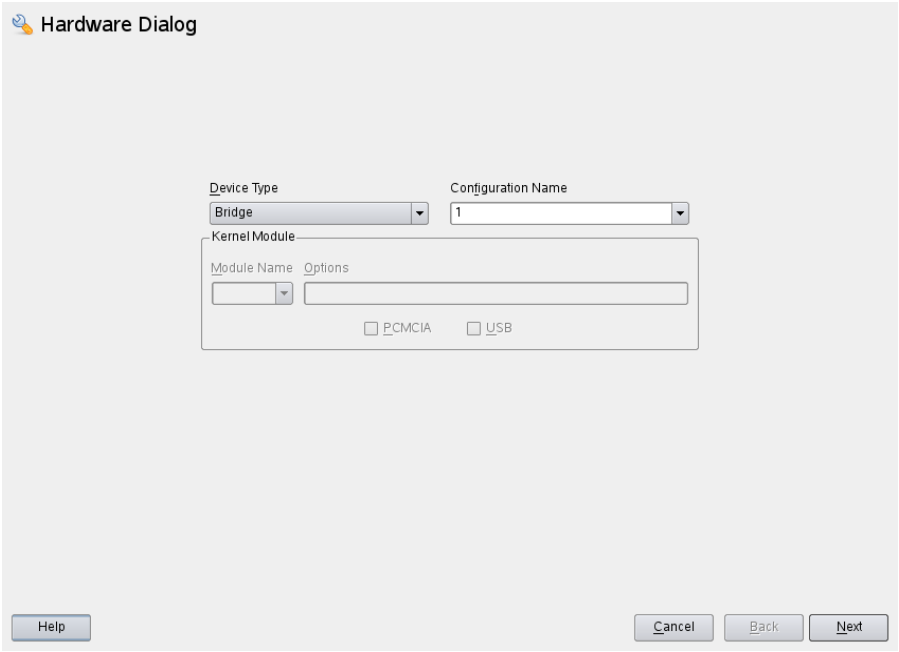
2.6 An Alternative Approach to Networking: Using Several Bridges

In the preceding procedure, you've seen how you can create a network configuration focused on the host computer. You've set up a bridge that used to connect all virtual machines to the network. That also means that no priority has been configured between the different machines. There is an alternative: you can create several bridges so that virtual machines can be assigned to specific network cards. This makes it easier to set up load balancing between machines that need a lot of bandwidth and machine that can function with less bandwidth. If required, you can even create a bond device within the virtual machine, connecting two of the virtual network cards together. To do this, you can follow the procedure for the creation of a bonding device that has been described above. In the next procedure, we'll create more than one network bridge on the host computer.

1. While installing the server, you can simply provide an IP address for each of the network cards. There's no need to consider the bridges you want to create later.

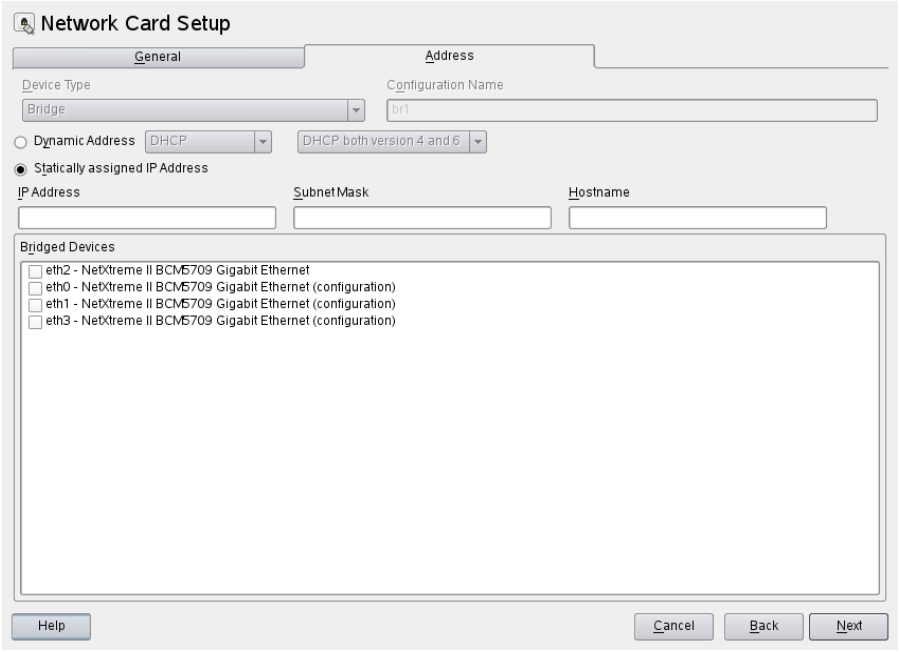
2. After configuring the server, start YaST. From YaST, select Network Devices > Network Settings. You'll now see the Overview window, with an overview of all the network connections that have been created. From here, click Add to add a new network connection.

3. In the Hardware Dialog window, from the drop-down list, select the Device Type Bridge. Don't worry about the configuration name, the first available number will be selected automatically. After selecting this, click Next to continue.



When creating more than one bridge, YaST ensures that each bridge gets the correct device number.

4. Now add a static IP address for your bridge, with the subnetmask you want to use. Then choose all the network cards you want to assign to the bridge. Make sure you don't select a network card that is used elsewhere, as this will cause problems!



Add a static IP address to your bridge and assign all network cards you want to use in the bridge.

5. Repeat this procedure until you have created all the bridges you need on each host.

3 Creating the Basic Cluster Configuration

Now that the network cards have been configured, it's time to take care of the cluster. To configure a High Availability cluster in a SLES 11 environment, you need the SLES High Availability Extension (HAE). This is a product that is sold seperataly from SLES 11. That means you have to pay an additional price for software that is normally available for free (see www.clusterlabs.org). In exchange for the money you pay, you get supported software with a complete ecosystem that is created to get you up and running as soon as possible, should you encounter problems. It is recommended to use the supported option if you're creating the cluster in an enterprise environment, but not required. In this procedure I'm assuming that you're using SLES 11 with the HAE though. So make sure that you've downloaded the software from http://www.novell.com/download and made it available to your SLES servers. You need to perform the following steps, which are described in this and subsequent chapters.

1. Install the Cluster Software
2. Create the basic cluster configuration
3. Set up STONITH
4. Create resources for the virtual machine storage backend
5. Create virtual machines as cluster resources

3.1 Installing the Cluster Software

Before you do anything else, you need to make sure that the cluster software is installed. The following procedure describes how to do this using YaST.

1. Make the installation media available to your server. For instance, put the installation CD in the optical drive of your server.

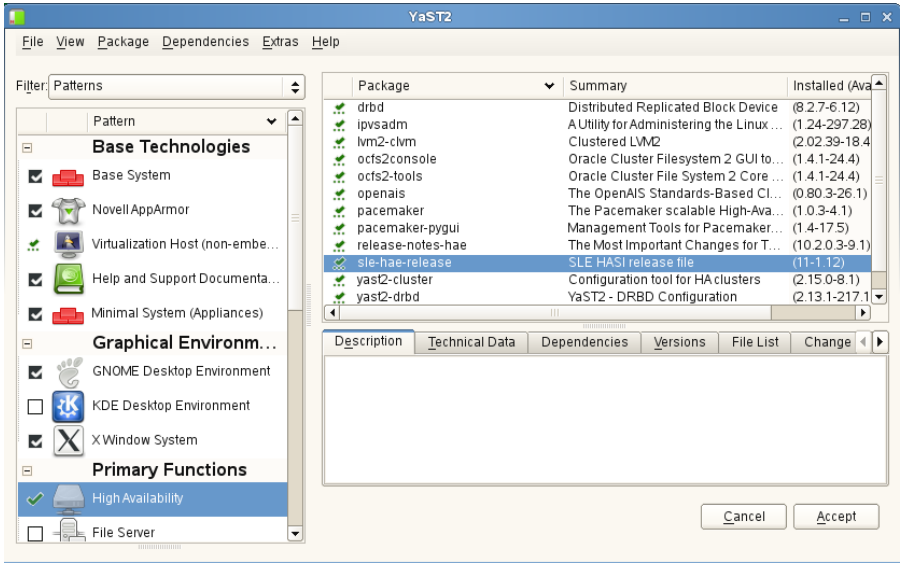
2. From YaST, select Software > Add on Products.

3. Click Add, select the option CD and then click Next to proceed.

4. Read the license agreement (if you don't have anything else to do) and click Accept to indicate that you agree to the conditions. Next, from YaST, select the software category High Availability and click Details to have a look at the available packages. The following packages are offered:

    - DRBD: This is the Distributed Replicated Block Device, a RAID like mirroring solution over the network. Very useful if you don't have a SAN, but not the best solution in this configuration.
    - ipvsadm:The IP Virtual Server, a network based load balancer.
    - LVM2: The Cluster aware Logical Volume manager (cLVM), which allows you to make logical volumes available to nodes on the cluster.

- OCFS2: The OCFS2 cluster-aware file system. This file system allows you to have write access from multiple nodes to the same file system simultaneously.
- OpenAIS and Pacemaker: The core cluster components that are required to create the solution described in this book.



Install all packages you need from the High Availability Extension

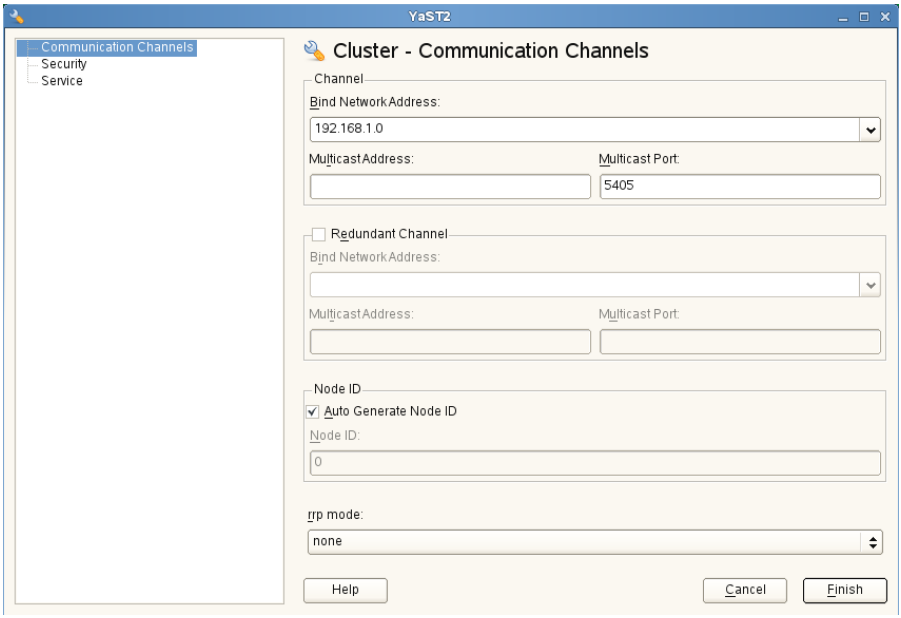
5. Now click Accept to confirm your choice. When asked if you want to install the software dependencies, click Continue to confirm.

6. At the end of the procedure you are asked if you want to configure the Novell Customer Center. Performing this step allows you to register the HAE product for your server and entitles you to receiving updates. It is a good idea to register your server now, although not required to continue the installation procedure. If you want to register now, make sure that you have the mail address that is used for your Novell account, as well as the registration codes for the product. With this information, you can follow the prompts to register your server and complete the installation procedure.

3.2 Connecting the nodes

Now that the software packages are installed, you can create the basic configuration for the cluster. In this initial part of the procedure, you'll ensure that the nodes are able to see each other and communicate with one another. The following procedure describes how to do this.

1. From YaST, select Miscellaneous > Cluster. You'll now see the main window from which you'll create the cluster configuration.



From this window, you'll create the initial cluster configuration

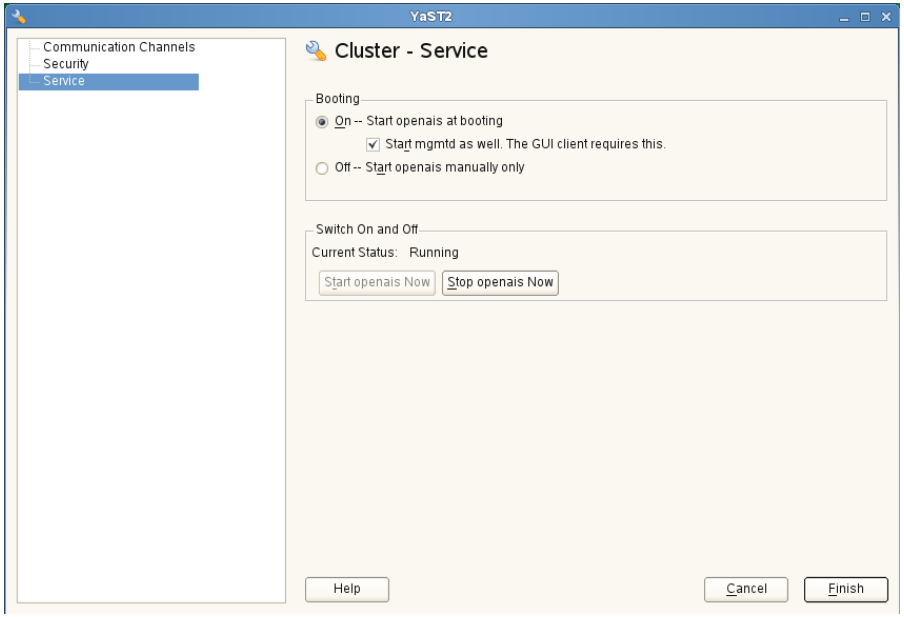
2. To start with, from the drop-down list, click the option Bind Network Address. This option is used to specify which network card to use for the cluster packets. The software will propose automatically all network addresses that are available on your server.
   For best performance, it's a good idea that you select a network interface that is representative for the user experience, hence you can select the network where the LAN traffic is sent on. After selecting this network, you need to specify a multicast address that will be used for the communication in the cluster. You can use any IP address that has a starting number between 224 and 239, as long as it's not already in use for something else. On other nodes you will use the exact same multicast address; the multicast address is used as a group address

to which all nodes in the cluster will listen. In this example configuration we'll use the multicast address 224.13.14.15.

Note: *A different approach for the cluster network interface, is to use a dedicated interface for the cluster. This offers the advantage that the normal traffic on the LAN will not interfere with the cluster traffic. However, it also involves the risk that an interface goes down without you knowing about it. To prevent this interface, you can configure the pingd resource. This cluster resource is configured to send ping packets to a node in the cluster on a regular basis. Using your default gateway would be an excellent choice for that. In case this ping host goes down, the cluster will know that there is a communications problem and fail over resources from the node that cannot ping any longer. Configuring a ping host is discussed later in this book.*

3. Before you continue, you need to decide if you want to configure a redundant channel for communication of cluster packets. The advantage of using a redundant communications link, is that the cluster packets will still be transmitted should the primary channel goes down. If however the primary channel is also used for the normal user traffic, you probably don't want the cluster traffic to fail over to the redundant channel without any action being taken on the cluster. This is a good reason why you wouldn't want to configure a redundant channel. If you do use the redundant channel, you'll need to set up a ping host as well, so that the cluster will know if an important connection in the cluster goes down. .

4. Now activate the Service option. From here, under Booting, click the radio-button On. This ensures that the cluster is added to all runlevels on your computer, which will start automatically. After that, click Start OpenAIS to star the cluster. Next, click Finish to save and apply your changes.



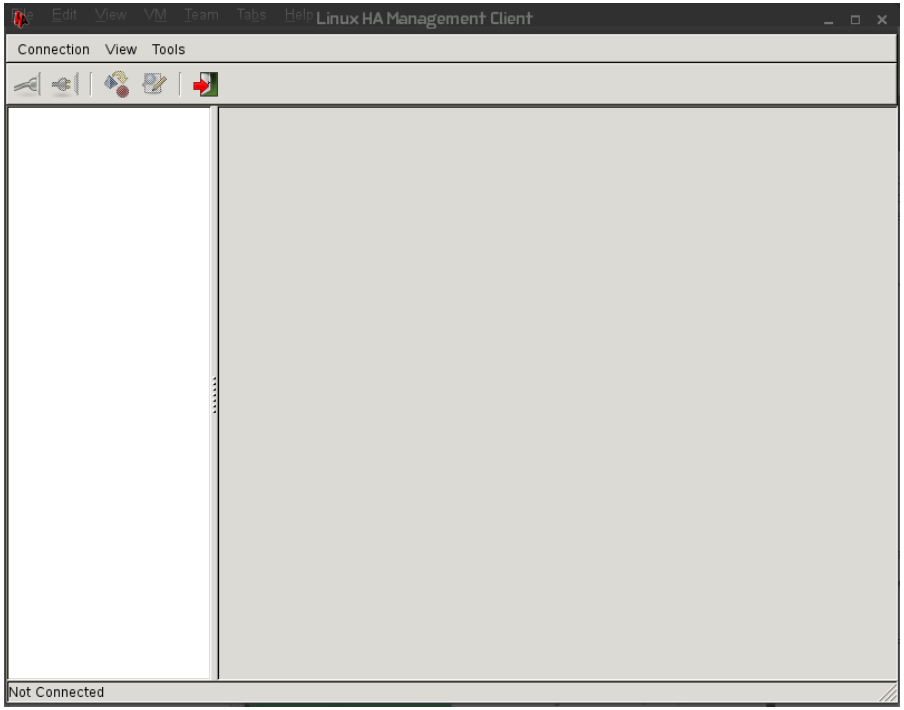
Indicate that the cluster has to be added to your runlevels and you want to start it now.

5. Congratulations, the cluster software is now activated on the first node. Repeat this procedure on all the other nodes that are a part of the cluster. Just to avoid any confusion about this: these are all the nodes on which you want to create the virtual machines.
   Once you've finished configuring the cluster software on all nodes, you can run the command crm_mon -i1 1 on one of the nodes. The crm_mon tool is the Cluster Resource Manager Monitor, which monitors what's happening in the cluster resource manager (CRM) - the heart of the cluster. This CRM decides on which nodes resources need to be started and using the command crm_mon -i 1, you request its current status, with a one second interval. The result should look as follows.

Listing: Use crm_mon -i 1 to monitor the current state of the cluster.

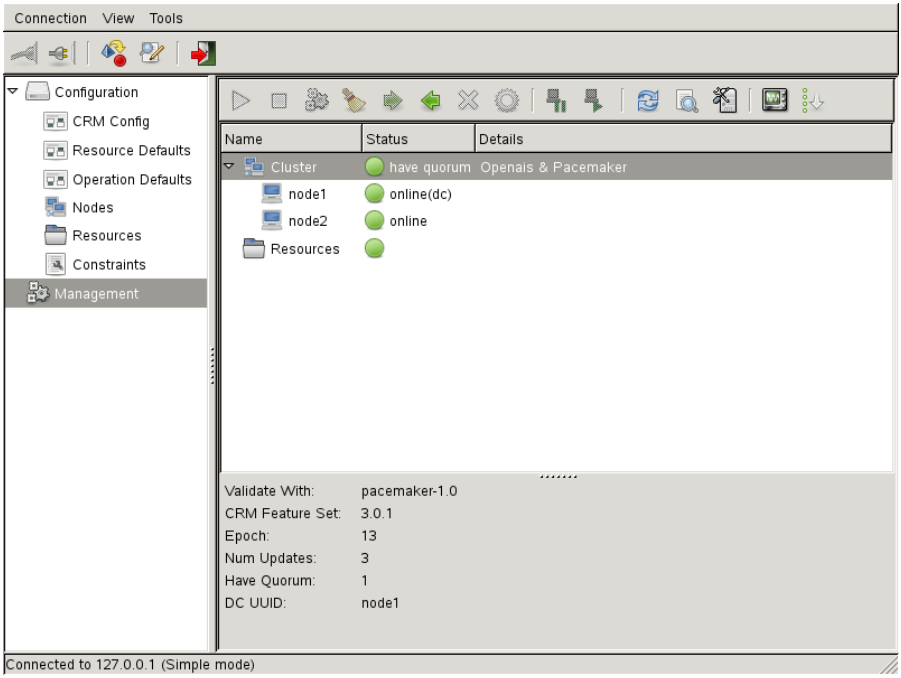
```
============
Last updated: Fri Dec 18 13:37:34 2009
Current DC: node1 - partition with quorum
Version: 1.0.3-0080ec086ae9c20ad5c4c3562000c0ad68374f0a
2 Nodes configured, 2 expected votes
0 Resources configured.
============

Online: [ node1 node2 ]
```

6. In the next step you need to prepare the graphical management tool for the cluster, hb_gui. This tool needs a user hacluster, which is created for you on each node where you have installed the cluster software. If you give this user a password, you can log in as this user to create and manage resources in the cluster. Hence, you need to use the command passwd hacluster to add a password for this user. As an alternative, you can also add an existing user account to the group haclient, which will give this user management rights in the cluster. Once you've done that, you can start the cluster management tool, using the hb_gui command from the console. This will show the window that you see in the following figure:



The initial hb_gui window

7. To manage and monitor the cluster from the hb_gui interface, you have to authenticate the user. To do this, click Connection, select the Login option and enter the password you have just created for the user hacluster - or authenticate as the user account that you have just added to the haclient group. It will take a few seconds before hb_gui has read all the information from the cluster. Once this has been done, you'll see a window giving you the current state of the cluster, as below:



After logging in to hb_gui, you'll see a window showing you the current state of the cluster.

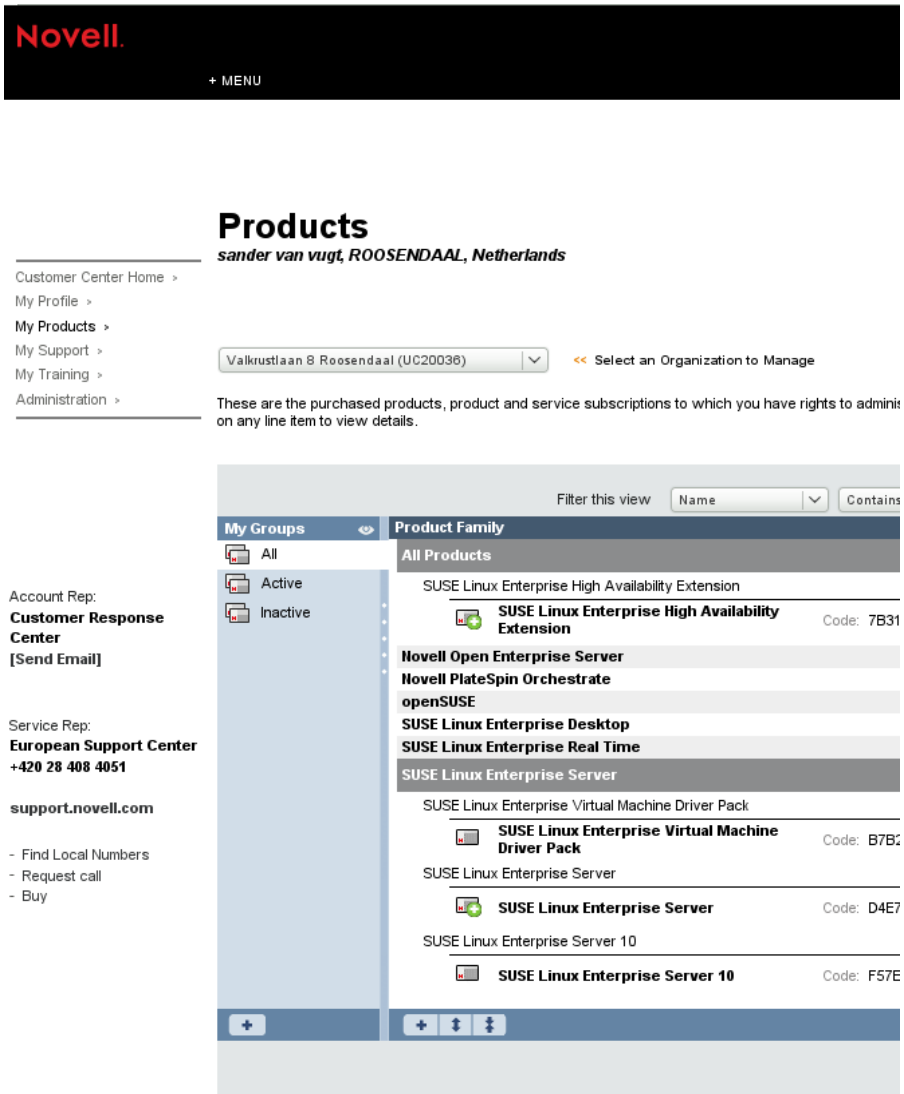
3.2.1 Registering and Updating your Server

After completing the software part of the installation of the HAE, you are prompted to register your server. In case you haven't done that before, here you can read how to register a SLES server and additional products that are installed on it. To register the server, you can follow the prompts at the end of the software installation. Alternatively, you can start the Customer Center program from YaST > Miscellaneous yourself. The following procedure describes how to proceed.

1. From a browser window , go to www.novell.com, log in using your Novell account and go to the Novell Customer Center; you can find an entry to the Novell customer center from the menu that shows on top of the Novell web site page.

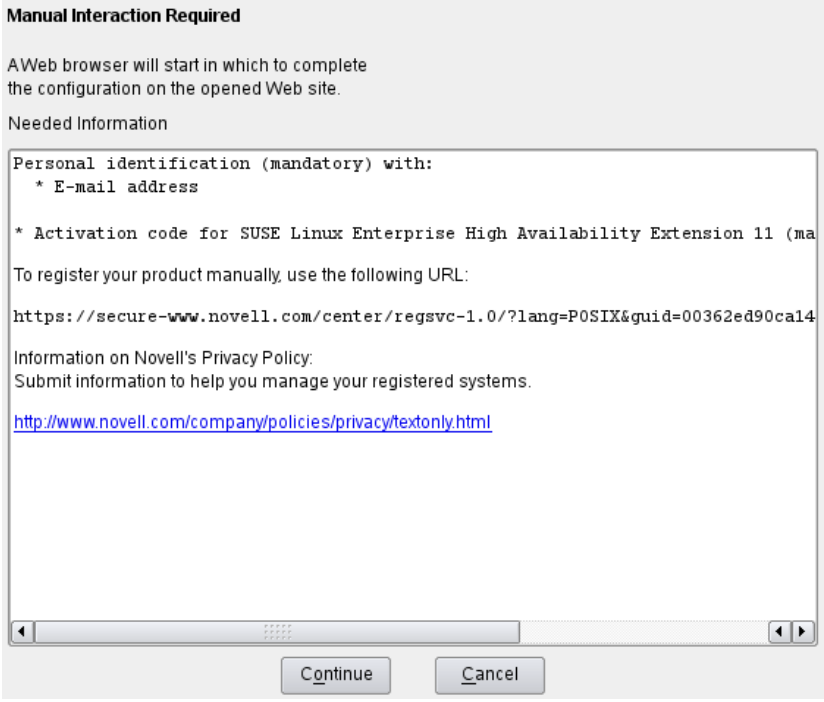
2. Select My Products > All Products and locate the SLES and HAE products that you've purchased. In here you see the registration codes that you need for your products.

***Tip**: You don't have to purchase the products to evaluate them. When downloading the software, you can request a free evaluation code. This evaluation code also shows in the customer center, so this procedure also applies if you only have an evaluation code.*



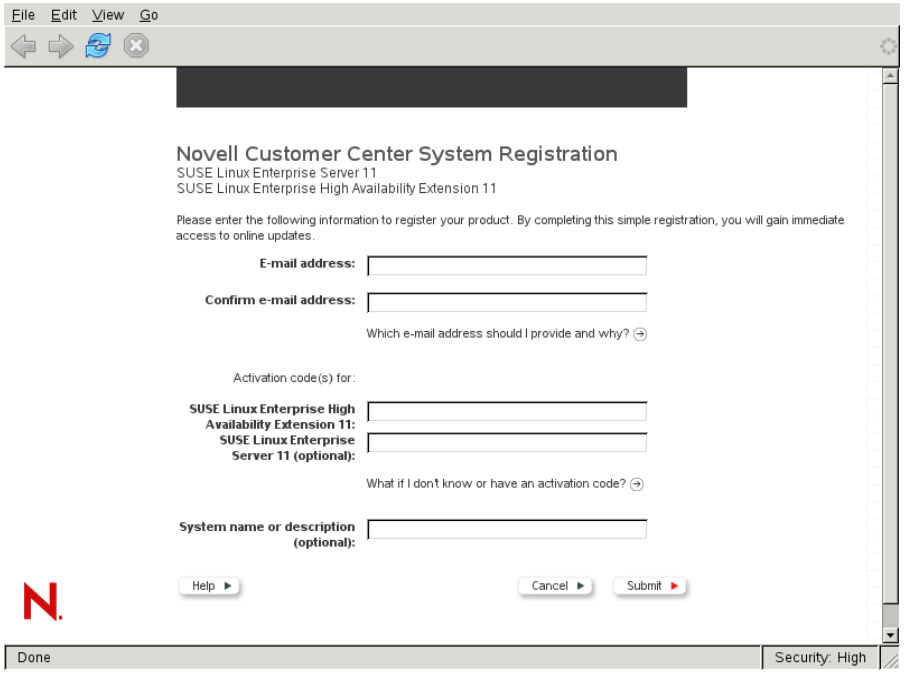
You can find the registration codes you need on the Novell Customer Center.

3. On the server you want to register, start YaST > Miscellaneous > Customer Center Setup. This option will try to contact the Novell registration server. When this has succeeded, you'll see a window as in the figure below. In this window, click Next to proceed.



After contacting the registration server succesfully, you'll see this window

4. From YaST, a browser window is loaded. In the browser you need to enter the registration credentials. You'll need the registration codes you've just found on the Internet, as well as the e-mail address connected to your Novell account. Enter this information and click Submit to proceed.



Enter your e-mail address and the registration codes for your products to proceed.

5. From the generic overview window, click Proceed to register your server. This will copy required credentials to your server, completing the registration . You'll now see that the server has been registered successfully. From the window confirming this, click OK.

6. At this point, you can update your server. Open a root console on the server, and from that console type the zypper up command. This will patch your server with the most recent software. Stay with your server, as you may have to reply to a confirmation prompt occasionally.

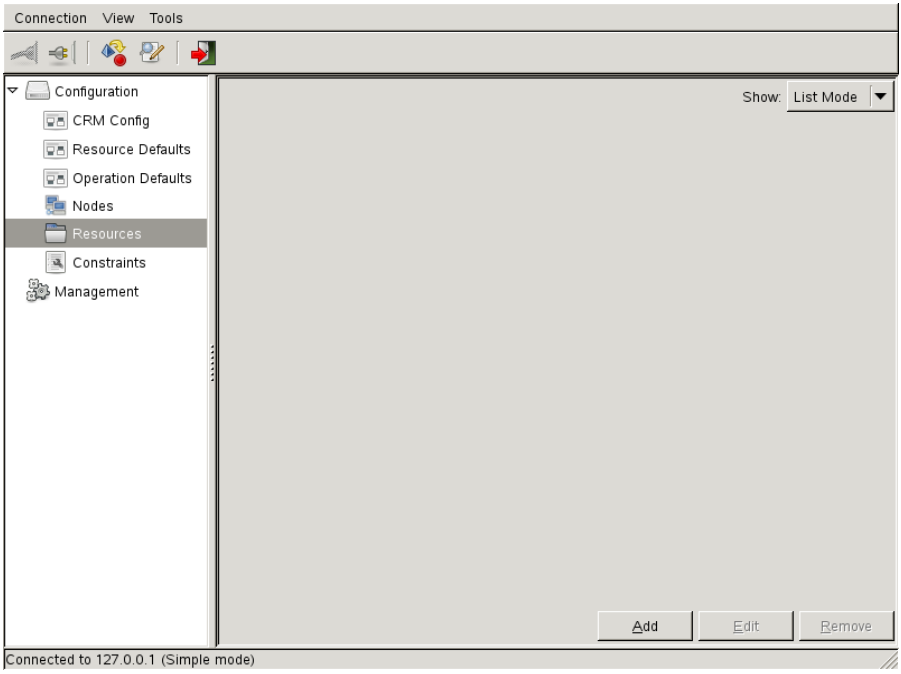
3.3 Setting up a Pingd Resource

If you've set up your cluster to communicate on a dedicated link, it's a good idea to configure a resource for a ping host in your cluster. The cluster will send ping packets to that dedicated host periodically, which confirms communication within your network.

The idea behind setting up a ping host, using the pingd resource, is each node in the cluster pinging an external host on a regular basis. You don't want a shell script that pings regularly, you want the cluster to send ping packets. To do that, you need a

resource in the cluster. There are two types of resources, primitives and clone resources. In fact, there are some other special purpose resources as well, but we'll ignore them for the moment. A primitive resource is started on one node only, a clone resource is started on multiple nodes simultaneously. To configure pingd, you'll need a clone resource, because every single node in the cluster needs to confirm that it is still capable of communicating with the rest of the network. The procedure below explains how to set up a pingd resource.

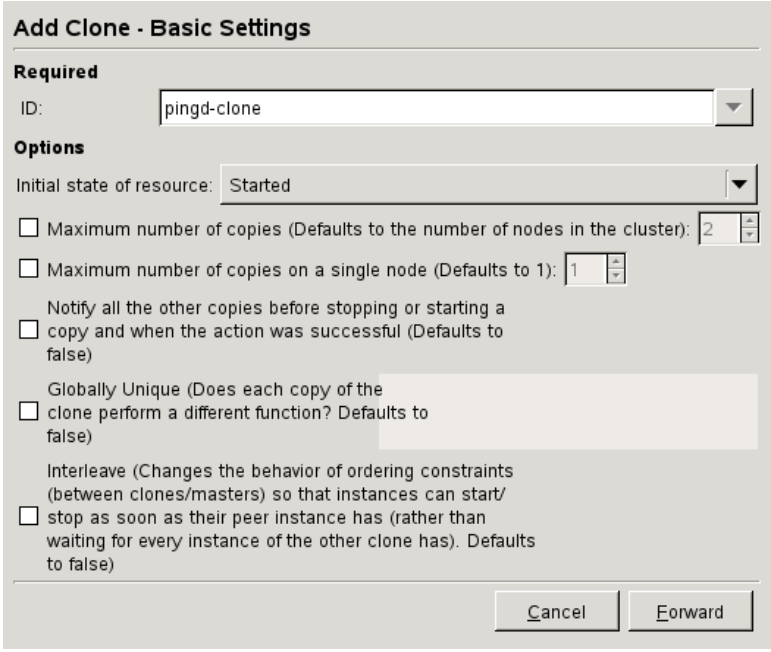
1. Make sure that the Linux HA Management Client is started. From there, select Configuration > Resources and click Add to add a new resource.



Click Add to add a new resource to the cluster

2. From the Add menu, select Clone to add a clone resource and click OK.

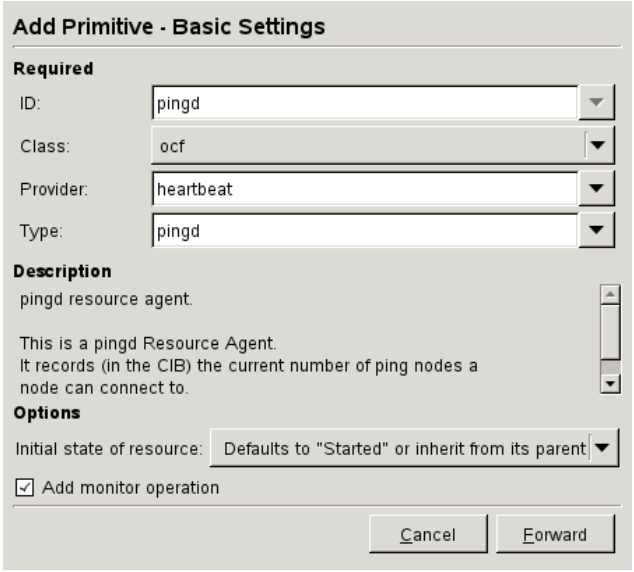
3. In the Add Clone - Basic Settings window, enter a unique name for the clone resource. In this example, we'll use pingd-clone as the name. Also make sure that the Initial state of the resource is set to Started and then click Forward to proceed.



Enter a name for the clone resource and click Forward

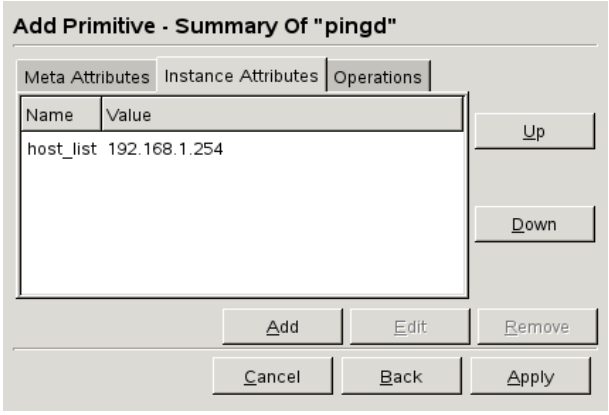
4. Now enter “primitive” and click OK to continue.

5. In the Add Primitive window, enter the ID “pingd”. From the Class drop-down list, select ocf, choose the heartbeat provider and next from the Type drop-down list, make sure that pingd is selected. Next click Forward to continue.



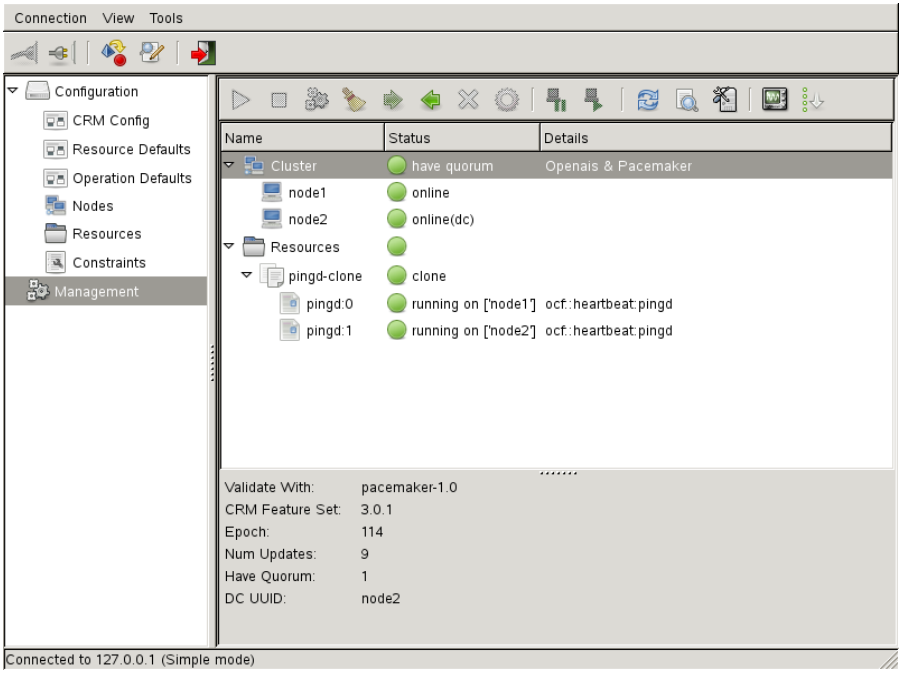
Creating the pingd resource

6. In the final step, you need to add the Instance Attributes. This attributes tells the resource how to do its work. For pingd, you need one attribute only, which is the host_list attribute that contains the IP address of the host you want to ping. To add it, make sure the Instance Attributes window is selected and click Add. From the Name drop-down list, select host_list and make sure to give it as its value the IP address of the host you want to ping. Then click OK to save it. This shows you the following window.



The pingd resource needs to know what host to ping

7. Click Apply twice to save the configuration. This brings you back to the main screen of the Linux HA Management Client. From this screen, in the left pane, click Management. This will show you that the pingd clone is up and running on all nodes in your cluster.



This is what it looks like if you've configured the pingd resource successfully.

4 Setting up STONITH to ensure resource integrity

Before you set up cluster resources in the cluster with access to shared storage, there is an important precautionary measure you have to take to prevent the cluster from getting corrupted. The name of the mechanism is STONITH and it stands for Shoot The Other Node In The Head.

In a HA cluster, it is of great importance to be sure that a node is down before taking over resources from that node. This decreases the risk of file system corruption to a minimum. When two nodes try to write to the same file system at the same time, you can be sure that your file system will get corrupted and bad things will happen. To prevent these kinds of situations, you need STONITH.

The idea behind STONITH is self explanatory. Before a resource or service is migrated from a node that is supposed to have a problem, STONITH makes sure that this node really is down. To do this, STONITH will simply shut it down. Typically, this is a hardware mechanism. Therefor to set up STONITH properly, you'll need a hardware device, such as a Power Distribution Unit (PDU) which has manageable ports that allows you to shut down the power on one of the ports. Alternatively, you could use a management card in your server, such as Dell DRAC, or HP ILO. To talk to this hardware, the cluster uses a STONITH resource agent that sends a command to the STONITH device to shut it down.
When configuring STONITH, the most important item to consider, is the device you use. Numerous devices are available, but not all are supported. In this book, we'll explore five different options:

- The APC PDU device. This is a power distribution unit that you can manage over the network. It provides the option to shut down a port remotely.
- The Dell DRAC management board. This is a management board that you have to install in your Dell server. Once installed, it allows you to send a shutdown command to the server.
- The SSH Stonith resource. This is a test resource that helps you in understanding how to use STONITH, but which should never be used in a production environment.
- The Meatware STONITH device. If you're in a production environment and you don't have the appropriate hardware, use the Meatware device. It won't do anything, but it will prevent file system corruption to occur on your cluster.
- The Null device. If everything else fails, the null STONITH device is a good choice. It doesn't do anything at all, but it keeps the cluster happy. Some resources need STONITH to be available and that's all that the Null device is doing.

4.1 STONITH or cLVM?

In the old days of Heartbeat clustering, there was only one method of preventing file system corruption and that was STONITH. In the modern days of Pacemaker, another approach has been made available: cLVM. In cLVM, you can mark a logical volume as private, which means that it is accessible by one server at a time. If one server has it mounted, others cannot even see it. That's also a way to prevent file system corruption.

If you have cLVM, you can do without STONITH. Some parts of the cluster do need STONITH software to be installed on your server, so you probably need both of them installed. The good news is that if you have a problem configuring STONITH (which is possible, because of different hardware versions there are some serious problems with STONITH), you can get away with a dummy STONITH device, if and only if you use cLVM for file system protection. It's always better to have real STONITH though.

In this book you can read about two different approaches to set up the shared storage. One of them uses cLVM. In case you decide to set up that solution, you don't really need STONITH. If however you decide to go for the solution where shared storage is offered from the OCFS2 file system (see Chapter 6), you will need to set up STONITH anyway.

4.2 Setting up STONITH on the APC Master

My personal favorite STONITH device, is the APC Master, sold as the switched rack Power Distribution Unit. When using this device on your network, you would connect the power plugs of all the servers to the APC Master device. The benefit of using this device, is that you can remotely power-cycle any port on the device. APC gives you a web interface or telnet to do this, with Pacemaker you can use a STONITH resource to accomplish the same.

Before using the APC Master in your cluster, you need to set it up. You can use the old-school method, using a serial cable and Hyperterminal on Windows, or you can use the following extremely cool procedure to give the device an IP address. Before you start, make sure that the APC is connected to your LAN, and that you have some servers connected to the power outlets of the APC.

1. Write down the MAC address of the device. You can find this address on a sticker on the device.

2. Use ARP to define a MAC address for the device on your local computer. On Linux, you would do that using the following command:

```
arp -s 192.168.1.245 00:c0:b7:4b:c9:d9
```

3. Now use ping with a package size of 113 bytes to set the IP address on the local device as well:

```
ping 192.168.1.245 -s 113
```

   You will see the device answering to the ping package.

4. Use telnet to connect to the device. The default username is apc, the default password is also apc. Make sure to change the password to prevent others from having fun with your equipment.

5. Next, choose Network from the Console menu, select TCP/IP and next Manual boot mode. You need this to tell the device that it's not booting through a DHCP server.

6. Back in the main Network menu, specify the System IP, Subnet Mask and Default Gateway address.

7. Use Ctrl-C to exit the Control Console menu, followed by option 4 to log out. This will write the changes to the device and make them persistent.

At this point, your APC device is ready for use in the cluster. You shouldn't believe things that you haven't seen yourself, so let's do a small test. The following procedure describes how you can power cycle a port on the APC.

1. Open a telnet session to the APC and enter the user name and password that you've provided for the device (default for both is apc). This gives you the main menu which you can see in listing X.

   Listing: The main menu from which you can control access to the APC PDU

```
American Power Conversion               Network Management Card AOS
v3.7.0

(c) Copyright 2008 All Rights Reserved  Rack PDU APP
v3.7.0

-------------------------------------------------------------------------------
--

Name      : RackPDU                                   Date : 09/19/2000

Contact   : Unknown                                   Time : 12:20:01

Location  : Unknown                                   User : Administrator

Up Time   : 0 Days 0 Hours 5 Minutes                  Stat : P+ N+ A+


Switched Rack PDU: Communication Established
```

```
------- Control Console
-------------------------------------------------------

     1- Device Manager

     2- Network

     3- System

     4- Logout

     <ESC>- Main Menu, <ENTER>- Refresh, <CTRL-L>- Event Log
```

2. From the menu, select 1 to get access to the device manager. This gives you access to three different options, from which you select option 2, Outlet Management.

3. At this point, select option 1, Outlet Control/Configuration. This gives you a list of power outlets and their current status. (See listing)

   Listing: The Outlet Control/Configuration menu gives you a list of all available outlets and their current status.

```
------- Outlet Control/Configuration
----------------------------------------


     1- Outlet 1                 ON

     2- Outlet 2                 ON

     3- Outlet 3                 ON

     4- Outlet 4                 ON

     5- Outlet 5                 ON

     6- Outlet 6                 ON

     7- Outlet 7                 ON

     8- Outlet 8                 ON

     9- Master Control/Configuration

     <ESC>- Back, <ENTER>- Refresh, <CTRL-L>- Event Log     >
```

4. Now select the outlet on which you want to shutdown a server, say that you want to work on outlet 1. Next choose option 1, Control Outlet.

5. At this point you see the menu with available options (see listing). From this menu select Immediate Reboot and confirm your choice by typing Yes. The power will now be recycled and your server will reboot.

Listing: The Control Outlet menu gives you different power management options

```
------- Control Outlet ------------------------------------------------------


        Name         : Outlet 1

        Outlet       : 1

        State        : ON


     1- Immediate On

     2- Immediate Off

     3- Immediate Reboot

     4- Delayed On

     5- Delayed Off

     6- Delayed Reboot

     7- Cancel
```

Now that you have a generic feeling of what you can do from the APC PDU, it's time to make it usable for your cluster. That means that you need to configure a name for each of the ports of the device. The cluster is going to talk to the APC, and tell it to switch down node1 for example. But in order to enable this, the APC needs to know what it's talking to when talking to node1. Configure this by giving a name to each of the ports on the device. After doing that, you also need to set it up for use of the Simple Network Management Protocol (SNMP). STONITH is going to use this protocol to talk to the device and to configure this, you need to set a password that allows SNMP to make changes to the current configuration. The next procedure describes how to perform these two steps.

1. Open a telnet session to the PDU and log in with the username and password that are set on the device (defaults are apc, apc).

2. From the main menu, select option 1, Device Manager. Next choose option 2, Outlet Management, followed by option 1, Outlet Control/Configuration. This gives access to the outlet configuration menu where you can enter a name for each of the outlets. Make sure that the name corresponds to the real host name of the node.

Listing: From the Outlet Control/Configuration menu, you can provide a name for each of the outlets on the PDU.

```
------- Outlet Management --------------------------------------------------

     1- Outlet Control/Configuration

     2- Outlet Restriction

     <ESC>- Back, <ENTER>- Refresh, <CTRL-L>- Event Log

> 1

------- Outlet Control/Configuration ---------------------------------------

     1- node1                    ON

     2- node2                    ON

     3- SAN                      ON

     4- Outlet 4                 ON

     5- Outlet 5                 ON

     6- Outlet 6                 ON

     7- Outlet 7                 ON

     8- Outlet 8                 ON

     9- Master Control/Configuration

     <ESC>- Back, <ENTER>- Refresh, <CTRL-L>- Event Log
```

3. After configuring a name for the outlet, make sure that you select option 5, Accept Changes to actually write the changes to the device. After doing that, press the Escape key five times, which brings you back to the main menu.

4. From the main menu, select Network and from the Network menu, select SNMP. In the SNMP menu, select 2 - SNMPv1 Specific Settings. This gives access to a list of four different access controls. Access control number 1 allows you to set the SNMP read community name, access control number 2 allows you to set the write community name. At this point, select option 2, which gives access to the default settings for the write community. It's a good idea to change the default setting where the community name private is used, to something more secure (anyone who uses “private” as the community name to access your PDU, will have complete write access to the device!

5. After making the changes, select option 4 to accept the changes and write them to the device. Next, press Escape until you get back to the main menu and log out from the device.

Listing: You should change the default write community name "private" to something that is more secure.

```
------- SNMPv1 Access Control 2 ---------------------------------------------


        Access Control Summary

        #  Community        Access         NMS IP

        ------------------------------------------------------------------------

        1  public           Read           0.0.0.0

        2  private          Write          0.0.0.0

        3  public2          Disabled       0.0.0.0

        4  private2         Disabled       0.0.0.0


     1- Community Name: private

     2- Access Type   : Write

     3- NMS IP/Name   : 0.0.0.0

     4- Accept Changes:


     ?- Help, <ESC>- Back, <ENTER>- Refresh, <CTRL-L>- Event Log
```

You have now used manual power cycling on a host, and you have set up the PDU to communicate to the cluster. In the next procedure you'll learn how to set up the cluster to use the PDU for STONITH operations.

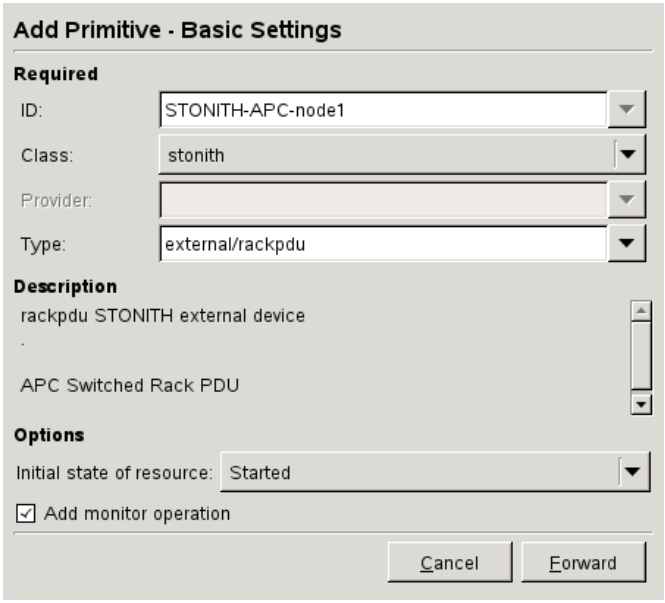
1. Open a console on one of your cluster nodes, and log in as the user hacluster.

2. Open Configuration > Resources and click Add to add a new resource. From the Add popup window, select Primitive and click OK.

Note: *For many STONITH resources, you would use clone resources that can run multiple instances on the cluster. For the rackpdu resource, this is not the case. The PDU can have one active connection only, so you should configure the rackpdu resource as a normal primitive resource.*

3. In the Add Primitive window, enter an ID for the STONITH device. You will create a resource for every node, so to STONITH node1, STONITH-APC-node1 would be a reasonable name.

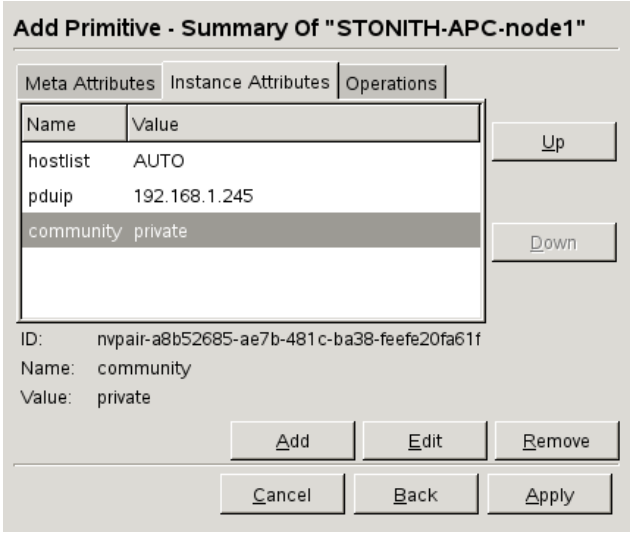
4. Still on the Add Primitive window, from the Class drop-down list, select stonith. Next, from the drop-down list at the Type option, select external/rackpdu. Be aware that the apcmaster does not work properly, so make sure you don't choose that! Also make sure that under Options, the Initial status of resource is set to Started. Your configuration should now look as in the figure below.



Configuring the APC Master Stonith Resource for a Node.

***Tip**: All external STONITH plugin modules exist in the file system of your server, in the directory /usr/lib[64]/stonith/plugins/external. Many of them are just scripts that are written in bash, and fairly easy to understand. So if you need to know more about the possibilities, browse to this directory and read how exactly the STONITH module has been programmed.*

5. Now click Forward to proceed to the next window. The first parameter you need to configure, is hostlist. It is advised setting this to AUTO, which means that the STONITH resource gets a list of host automatically from the PDU. Next, you have to specify the IP address of the PDU at the pduip parameter. The last required parameter, is the community name. This is the name of the SNMP community that you have specified on the PDU to manage it using the Simple Network Management Protocol.



You need to specify at least the IP address and SNMP community name of the PDU

6. At this point, your STONITH configuration should be working. You can test it by just killing the aisexec process on one of the servers.

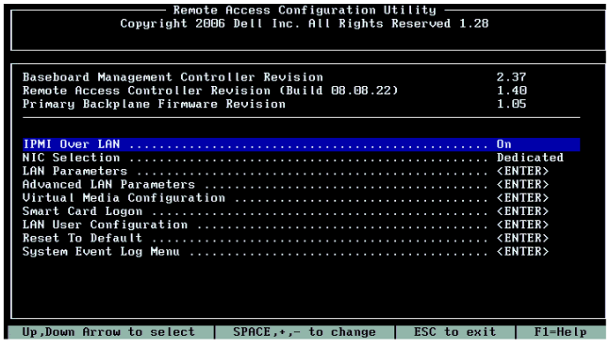
4.3 Configuring STONITH for Dell DRAC and other server management cards such as HP ILO

One of the devices for which it might be harder to configure STONITH, is for Dell's DRAC management interface. The problem with this interface is that Dell has released so many DRAC versions. The current release is DRAC6i, but because of the lack of devices to test, the DRAC STONITH agent never got beyond version 5. That means that you will have a hard time configuring STONITH for this device. The following procedure teaches you how to set it up anyway, hoping you have the right version of DRAC (or that an update has been made available by the time you read this).

***Note**: Because of the lack of the appropriate hardware in my test lab, I can cover DRAC only. The procedure described here works the same on other server management boards though, such as HP ILO. This card can be configured using the RILOE STONITH agent, and uses parameters that are comparable to the configuration you would use on DRAC.*

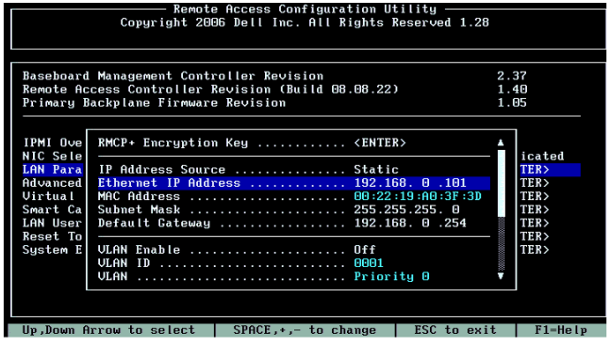
Before you can do any STONITH on a DRAC device, you need to set up it's BIOS. If you don't do that, the device won't have an IP address and there is no way for you to communicate to the device. To enter the DRAC BIOS, monitor the boot

procedure on your server. When on the server console you see the message about the Remote Access Configuration Utility, use the Ctrl-E key sequence. This brings you to the DRAC configuration menu, which you can see in the figure below.



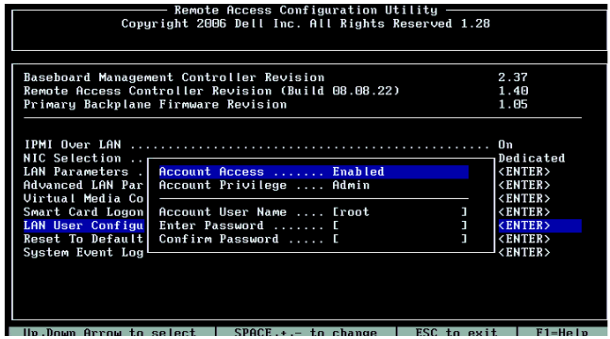
To specify the IP address for the DRAC card, you have to configure the DRAC BIOS.

Now choose the part of the menu where you have to enter the IP-address and make sure that the DRAC interface has it's own unique IP address. You must also make sure that this is an IP-address that you can reach from the console of your servers as well, as the cluster has to be able to communicate directly to the DRAC card to switch it off. If required, also think of configuring the required VLAN information.



Make sure that you give the DRAC card an IP address that you can also reach from your server's console.

After configuring the IP address, you need to create an administrator account. DRAC by default uses a user with the username root. Give this user a password. Before putting in your password, be aware that you will have to enter it in plain text in a configuration file that is used by the cluster, so don't use a complex password!



Provide a password for the DRAC user root

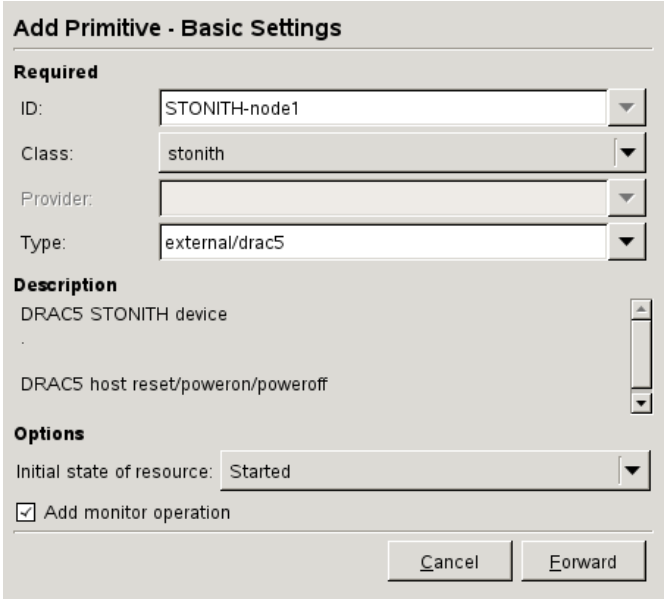
Note: *Dell sells two versions of DRAC, real DRAC and DRAC express, which uses a shared IP address (and also has no ability to remote control the server). DRAC express is useless in a Pacemaker environment, so make sure that you have the Enterprise version of DRAC, which can be configured with its own unique IP address.*

After configuring the DRAC BIOS, press Escape to exit the BIOS and boot the server. Once it is up and running again (and you have performed this procedure on all the nodes in your cluster), you can start the hb_gui and create STONITH resources for DRAC.

1. From the console of one of the host servers in the cluster, start the Heartbeat Management Client by typing the hb_gui command. Next, log in as user hacluster with the password that you have entered in the DRAC BIOS.

2. From the hb_gui, select Configuration > Resources and next click Add to add a new resource. From the Add window, select the resource type Primitive. For DRAC, you don't configure a clone resource as you need a DRAC resource for every single server in the network.

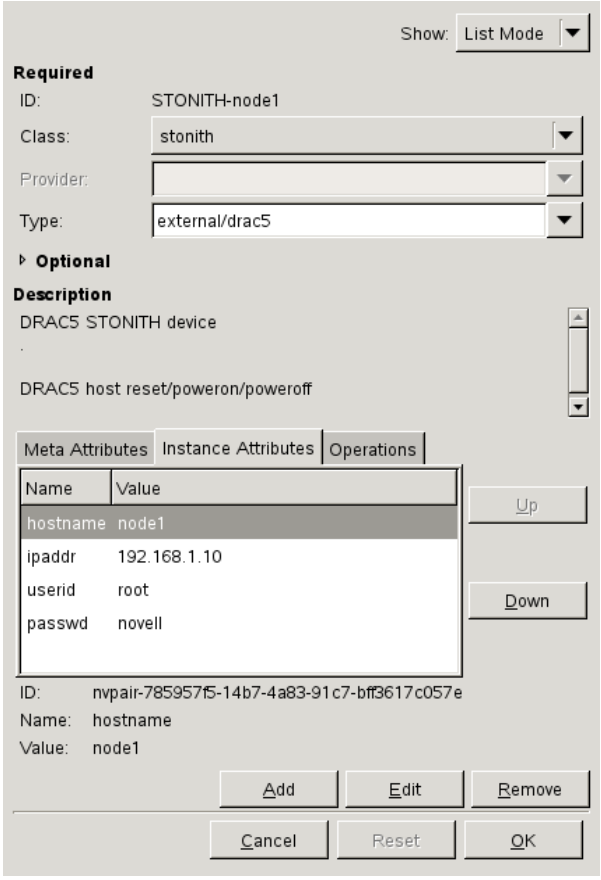
3. Now enter the required parameters. You need the following attributes:
   - ID: This is the name with which the resource is created in the cluster. The STONITH agent for node 1 for example, could have the name STONITH-node1.
   - Class: Here you have to select the resource class stonith.
   - Type: Select external/drac5 for DRAC5 and later. There is also a resource for drac3, but that is very old and probably not useable in many environments anymore.
     Also, under Options, set the initial status of the resource to Started.



Configuring the STONITH resource for DRAC

4. Now click Forward to continue. This brings you to a window where you can enter additional attributes for the resource that you want to configure. Select the Instance Attributes tab, and next click Add. Now configure the following attributes:

    - hostname: Here you put the name of the specific host that you want to manage with this STONITH resource.
    - ipaddr: Use the IP address of the device that you want to reach using STONITH. So this is the IP address of the actual DRAC card itself.
    - userid: In here, specify the name of a user who has permissions to use STONITH to switch off the server using DRAC (typically, this would be root).
    - passwd: Here you enter the clear text password of the administrator user id.
    - Click Add to save the configuration and add the DRAC resource to your cluster.



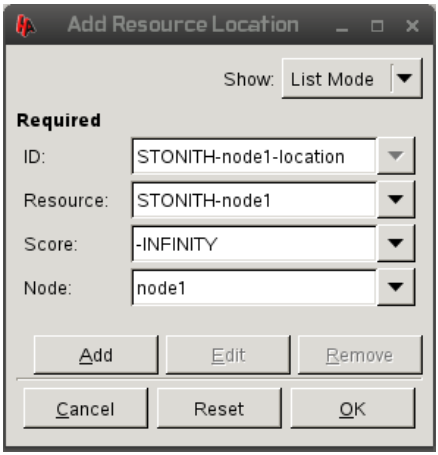
Specifying required properties of the STONITH device.

5. At this point, you need to make sure that the resource that you've just created can be started anywhere, but on the host that it has to shutdown. You do this by creating a location constraint, which is a rule that specifies where the resource is allowed to be started. From the hb_gui, select Configuration > Constraints and click Add to add a constraint. Then select the Resource Location type.

6. Enter a name for the constraint that allows you to find it easily. It is important to be clear about names, as you will end up with lots of resources in the cluster, which makes it hard to find the resource that you need. Next, enter the following three attributes:

    - Resource: From the drop-down list, at this point you select the resource for which you create the constraint. That would be STONITH-node1 for example.

- Score: The score is an indication of how serious you are about this constraint. By using the score -INFINITY (read “never”), you specify that the resource may be started anywhere, but on the node that you select in the next step.
- Node: Here you choose the node for which you have created the resource, which is the node that your DRAC resource is meant to shut down in case of problems.



The location constraint makes sure that the STONITH resource is started anywhere, but on the node that it has to shut down.

At this point, you have a DRAC STONITH resource operational on one node. You have to repeat this procedure for all the other nodes in the cluster as well.

4.3.1 SSH STONITH

Up to now we have seen two viable STONITH resources. You may however find yourself in a situation where a real STONITH device is not available. The problem is that the software discussed in this setup, doesn't work without a STONITH device. Hence, you might want to use a test resource. One of these test resources could be the SSH STONITH resource. In this subsection you'll learn how to configure it.

WARNING: *The SSH resource is not created for a production environment. In fact, it's a bad idea to use it at all, unless it's for your own test lab where you don't have real server hardware available. The risk with the SSH STONITH resource, is that you'll find yourself in some kind of STONITH ping-pong game, where you enter a STONITH loop, which means that servers will STONITH each other without interruption. So use this resource at your own risk!*

To configure SSH-STONITH, you first need to prepare the nodes in the cluster. You'll need host name resolving (if you don't have a DNS server that takes care of that), and you need to configure SSH for key-based authentication. Last, you have to start the atd process. The SSH STONITH agent can't work without this process, so make sure it is started. The steps below describe how to accomplish these tasks.

***Tip**: The atd process is also the solution to use if you find yourself in an SSH-STONITH-storm. Just log in to the server directly when it comes up, and get the atd process out of your runlevels using insserv -r atd.*

1. In an ideal world, you would take care of host name resolving using DNS. If you don't have a DNS server at hand, you'll need to create entries in the /etc/hosts file on all of your servers so that you can reach the nodes in your cluster by using their names. Make sure that you use the IP address that is used by the cluster software itself, not the IP address that you're using to connect to the SAN.

2. On all hosts, use ssh-keygen -t dsa to create an SSH key pair. This command will ask you a couple of questions. On all of these questions, you can just accept the default suggestions by pressing Enter. Also make sure that you don't enter a passphrase for the private key, because that would make it rather complicated to enable automatic SSH login in the cluster. The listing below gives an impression of the entire procedure of creating the SSH keys.

   Listing: Use ssh-keygen to create a key pair

```
node2:~/Desktop # ssh-keygen -t dsa
Generating public/private dsa key pair.
Enter file in which to save the key (/root/.ssh/id_dsa):
Created directory '/root/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_dsa.
Your public key has been saved in /root/.ssh/id_dsa.pub.
The key fingerprint is:
d5:3f:88:ff:f7:54:a1:e1:64:8f:57:0c:89:d3:a0:74 root@node2
The key's randomart image is:
+--[ DSA 1024]----+
|           . E+.. |
|          . +o oo |
|           o .= .o|
|          . .+o= o|
|        S . .+oo.|
|            .  ...|
|             .   .|
|              . ..|
|               ..o|
+-----------------+
```

3. Now you need to copy the public key from this host to the other hosts. To do this, you can use the ssh-copy-id command, followed by the name of the host where you want to copy the keys to. You'll need to do this for every host in the cluster. For instance, use ssh-copy-id node1 to copy the public key of this host

to the host that uses the name node1. In the listing below you can see what the result of this command looks like. After successfully copying the keys, you can now log in between two hosts without entering any passwords.

Listing: Use ssh-copy-id to copy the public key from one host to another

```
node2:~/Desktop # ssh-copy-id node1
The authenticity of host 'node1 (192.168.150.10)' can't be established.
RSA key fingerprint is 8e:4b:ed:2b:2a:15:34:5e:54:50:14:35:20:2c:29:b5.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'node1,192.168.150.10' (RSA) to the list of known
hosts.
Password:
Now try logging into the machine, with "ssh 'node1'", and check in:

  .ssh/authorized_keys

to make sure we haven't added extra keys that you weren't expecting.
```

4. Now you need to start the atd service and make sure it is started automatically from the runlevels. The following two commands will do this for you:

```
insserv atd
rcatd start
```

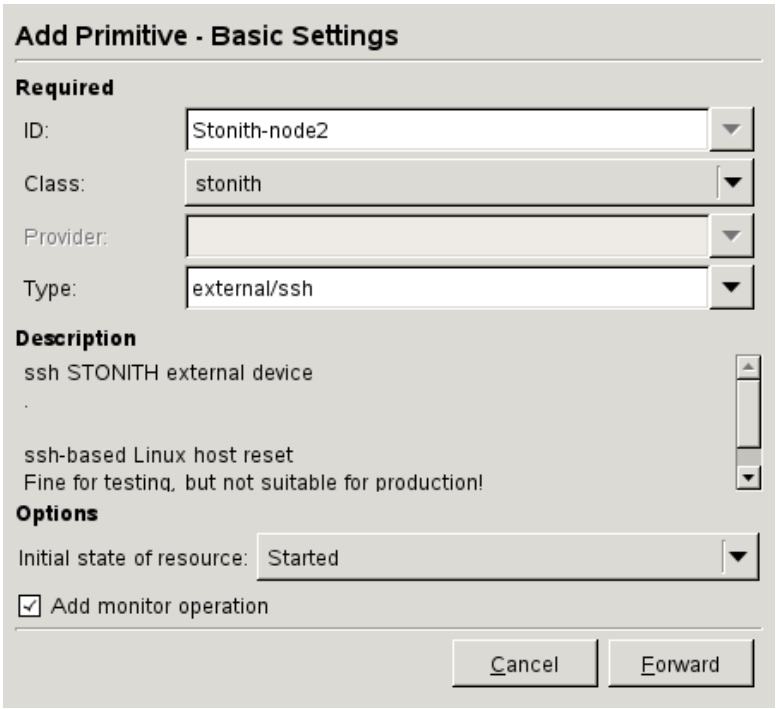
At this point you've done all of the preparation to use STONITH. Now you need to create the STONITH resources themselves in the cluster. To do this, you can use the hb_gui utility. So make sure that it is started and you have logged in using the appropriate credentials.

To use STONITH, you need a STONITH agent for every node in the cluster. This agent is responsible for shutting down a failing node. Stated otherwise, to make sure that node1 is shut down in case of problems, you need a STONITH resource somewhere in the cluster that can shut it down. Some people like to use clone resources for this purpose. As you have seen in the previous chapter, these are resources that can be started more than once in the cluster. The idea behind this configuration, is that every host in the cluster should be able to terminate a failing host. As this is useful only in a situation where you would expect multiple hosts to be failing at the same time, I don't use this configuration. In normal clusters you wouldn't have multiple nodes failing simultaneously.

Also, the fact that a resource is created as a primitive, doesn't mean that the resource is bound to one single host in the cluster. So if the STONITH resource for node1 was available on node2 and node2 fails, the STONITH resource would fail over to the next available node in the cluster. The only thing that you need to take care of, is that the STONITH resource is never started at the node that it needs to STONITH!
The procedure described below describes how on node1 you can create a STONITH resource that shuts down node2. To make sure that this resource is never started on node2 (suicide is not allowed), after creating the resource, a place constraint is created to prevent this.

1. In the hb_gui, under the option Configuration, select Resources and click Add to add a new resource. You'll now see a menu where you need to specify what type of resource you want to create. From this menu, select Primitive to create a "normal" resource.

2. In the Add Primitive window, you now have to specify how the resource is to be configured. To start with, give it a name, preferably one that makes it easy to recognize what exactly the resource is doing in your network. For instance, STONITH-node2 would be a nice name, as this resource is meant to shut down node2 in case of problems. At the class option, you select STONITH and next you specify the type External/SSH. Lastly, make sure that the Initial State of Resource is set to started and then click Forward to proceed and enter additional parameters.

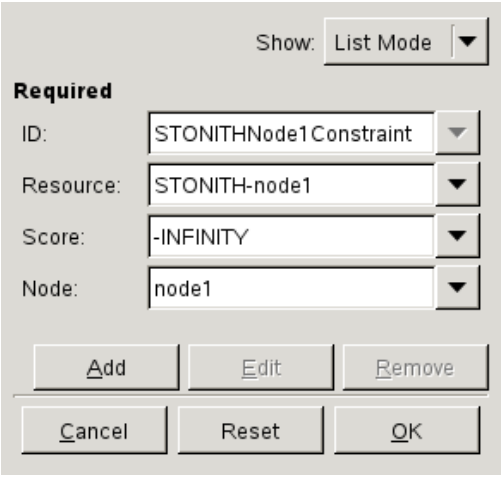


Creating the STONITH resource

3. You now see a window with three tabs. By default, the Instance Attributes tab is opened. On this tab, you see the hostlist attribute. Select this attribute, click Edit and enter the name of the host that is to be managed by this STONITH agent. Next click Apply to save the resource in the cluster.

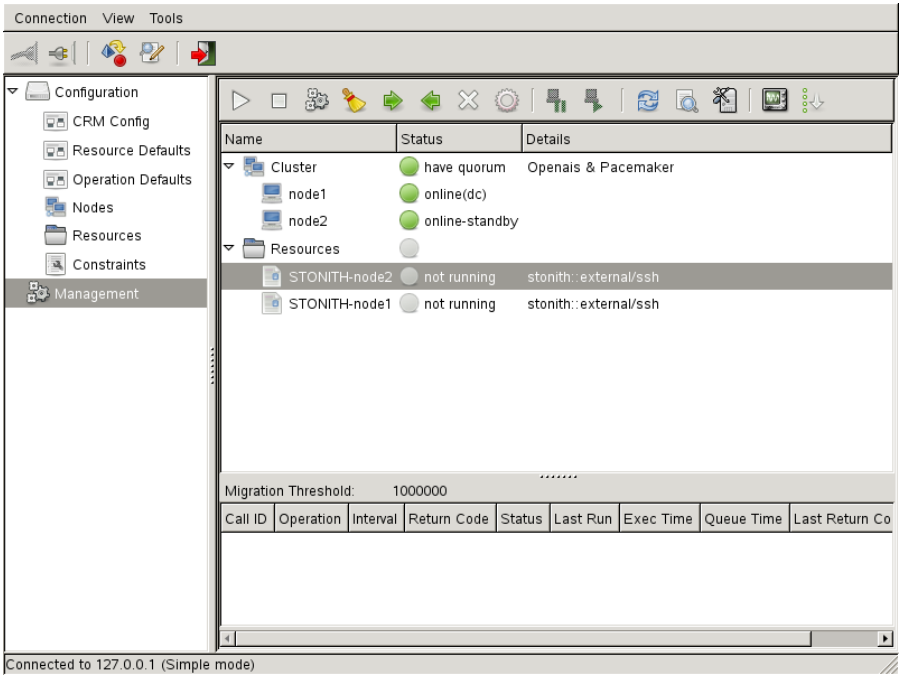
***Note**: If at this point you get an error that the resource cannot be written to the cluster, you probably havent' updated the software on your servers yet. Stop what you're doing immediately and update your servers now, before you do anything else!*

4. Now you need to make sure, that the STONITH agent that you've just created can be started on all hosts, except the host that it is supposed to shut down in case of problems. So the resource that you've created to shut down node1, must never be started on node1. To take care of that, you need a location constraint. By configuring this placement rule with a score of -INFINITY (read: never) for node1,
   you'll make sure that it will never be started on node1. To enable this constraint, from the hb_gui, select Configure > Constraints and click Add. From the pop-up window, select Resource location. As the ID, you have to provide a unique name for the resource. Next, from the drop-down list, you have to select the resource to which this constraint applies, which in this case is the SSH-STONINTH resource for node1. Next, give this resource the score -INFINITY and assign it to node1. Now click OK to save the resource and apply it. Perform the same steps for the resource that you've created to STONITH node2.



   By using a location constraint, you can make sure that the specific resource will never be started on the node it has to STONITH.

5. At this point, the SSH STONITH resources are created and available to the cluster. In the figure below you can see how it should look at this point. It's recommended to restart all nodes in the cluster so that the resources are loaded properly. Once you have done this and the resources have loaded, you can proceed to the next step where you create resources for the shared storage.



At this point the STONITH resources are ready for use

4.3.2 Configuring Meatware STONITH

In the last paragraph you've read how to set up STONITH based on SSH. This STONITH method will work, but there are some disadvantages as well. Therefore, you might prefer an alternative way to STONITH nodes in the cluster by using the Meatware device (yourself, as the system administrator). This STONITH agent does only one thing, it sends a message to the syslog to tell the administrator that he has to manually shut down a node. After doing that, the administrator has to confirm his action, by issuing the meatclient -c command. Only at that point, the cluster will know that everything is in a controlled state and it will migrate the resource. That means by using meatware, you don't have high availability at all! But the advantage is that it will prevent your resources from corrupting, as they will never be loaded twice. Also, you'll never find yourself in a STONITH loop, as is the case when using SSH based STONITH.

To use Meatware STONITH, you need a STONITH agent for each of the nodes in the cluster. The task of this agent is to shut down a specific node in case it fails. In other words, to shut down node 1, somewhere else in the cluster there must be a resource that can shut it down. So just as when configuring the SSH STONITH agent, you need a primitive resource that can run anywhere, but on the node that it has to shut down in case of problems.

The procedure below described how you can configure a Meatware STONITH resource on node1 to shut down node2 in case of problems. You'll also learn how to make sure that this resource will never be loaded on node2.

1. In the Linux HA Management Client, under the Configuration option, select Resources and click Add to add a new resource. You'll now see a menu in which you can specify which resource type you want to create. From this menu, select the Primitive option to create a normal resource.

2. From the Add Primitive window, you now have to specify how to configure the resource. To start with, you should give it a name. As always, it's a good idea to use a name that helps you in recognizing later for what purpose you have created the resource. So in this example, STONITH-node2 would ne a nice name. At the Class option, select STONITH and for Type, make sure that you have selected Meatware. Finally, under the options make sure that the Initials State of Resource is set to Started. Then click Forward to proceed to the next step where you can enter additional parameters.

3. You'll now see a window that consists of three different tabs. By default, the Instance Attributes tab is opened. On this tab, you can see the default attribute hostlist. Select this attribute, next click Edit and enter the name of the host that this STONITH resource has to manage. Next, click Apply to save this resource in the cluster.

4. Now you need to create a location constraint, which makes sure that the resource is allowed to start anywhere, but on the node that it has to shut down. So in this example, where we've created a resource to STONITH node1, the location constraint has to make sure the resource is never loaded on node1. To do this, you'll give the constraint the value -INFINITY. This is exactly the same procedure as has been described at the configuration of the SSH STONITH agent.

5. After saving the configuration, you'll have the meatware STONITH agents operational. You can now test if they really work the way you want them to. The easiest way to do that, is by killing the aisexec process on one of the nodes. You should see a STONITH message passing by on the other node, telling you that you manually have to confirm that the failing node is no longer available in the cluster. After doing that, by using the meatclient -c command, the cluster will know that everything is all right, and it will fail over resources.

4.3.3 If Everything Else Fails: NullStonith

You will find out soon enough, STONITH may be hard to configure. The problem however is that some configurations, including the configuration that is discussed in this book, need STONITH to be configured anyway. To help you with that, for situations where everything else fails, you can use the NullStonith resource. This

resource will not do anything at all, but the cluster will be happy anyway, as a STONITH resource is available. Notice that the Null Stonith device does not offer any protection, it will just lie to the cluster and pretends as if the STONITH operations have been executed properly. From that perspective, you might even consider it a worse resource that the SSH-STONITH solution. But, if it's the only way that can be used to achieve a working STONITH configuration, it's better than nothing.

To help you in setting up this resource, we'll use a different approach. Up to now, you have configured resources from the graphical user interface. As an alternative, you can also create an input file and import that in the cluster using the crm command. To use this method, first create a configuration file with the name nullstonithfile.txt and the content that you see in the listing below:

Listing: Input file to create a null-stonith resource.

```
configure
 primitive stonith-node1 stonith:null \
 params hostlist="node1"
 primitive stonith-node2 stonith:null \
 params hostlist="node2"
 location l-stonith-node1 stonith-node1 -inf: node1
 location l-stonith-node2 stonith-node2 -inf: node2
commit
```

Notice some variables in this example script that might need to be changed in your particular configuration. First, there is the name of the hosts, which in this example is set to node1 and node2. Change that to match you host names, and add lines if you have more than two nodes. Next, we've used the names stonith-node1 and stonith-node2 for the STONITH resources themselves, and l-stonith-node1 and l-stonith-node2 for the location constraints. Make sure that they are changed to match your environment. Next, you can import the configuration using the following command:

crm < nullstonithfile.txt

5 Setting up cLVM Shared Storage

At this point, all the preliminary work has been done and you can start configuring the resources that you need for the Xen high availability environment. The first part of the configuration, takes care of the storage for your environment. A virtual machine needs a virtual hard disk and this virtual hard disk has to be represented by something on the host computers. There are two approaches for this configuration.

Formerly, it was common to configure a cluster safe file system to take care of the storage back-end for virtual machines. In this configuration, OCFS2 (which is included in the SLES11 HAE) was used to provide one huge file system on which the storage back-end for virtual machine was taken care of by creating files to represent the virtual disk. There are however a few disadvantages to this approach:

- OCFS2 adds another layer of complexity to the configuration.
- Using a file as the storage back-end for a virtual machine is not as fast as using direct device access.
- Using one big OCFS2 volume adds a single point of failure to your configuration.
- In the past, OCFS2 was not stable enough, which in worst-case scenario's might mean that the entire cluster went down because of an OCFS problem.

Against all these disadvantages, there's also one advantages to using OCFS2: using files as the storage back-end is more flexible, as you can easily copy the disk file to another location. As we want to offer you information to set up both solutions, in this chapter you can read how to set up a cLVM based solution, whereas the next chapter explains how to set up shared storage based on OCFS2.

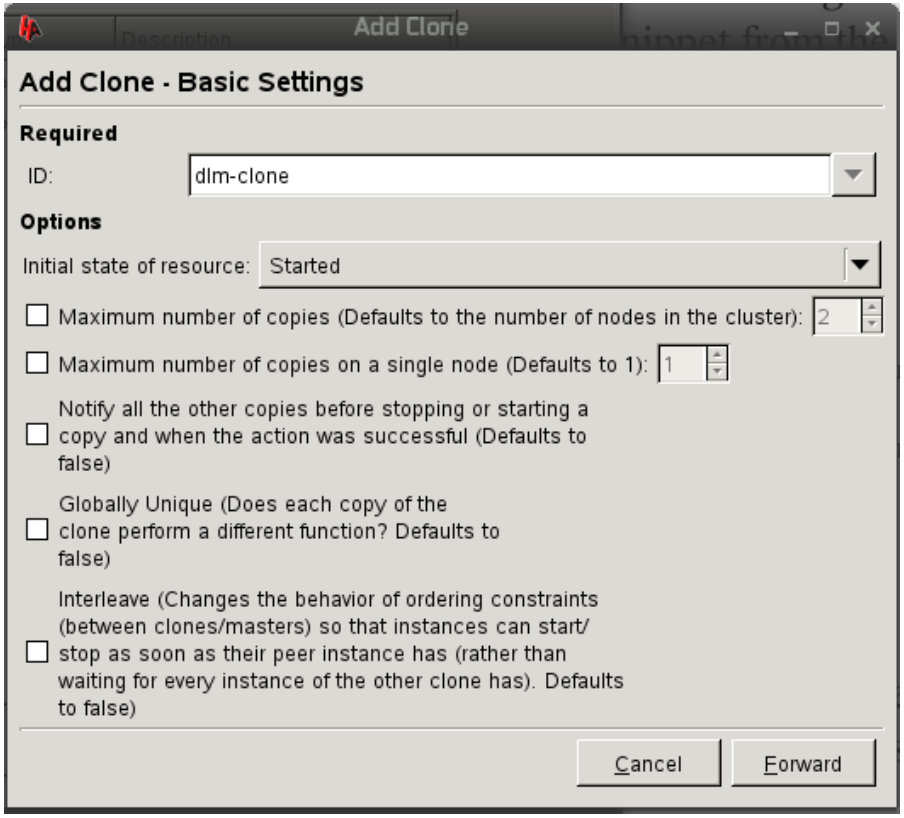
The alternative to using OCFS2, is the Cluster Logical Volume Manager (cLVM). In this configuration, you'll create logical volumes that can be managed by the cluster. That means that the cluster decides who has access to a volume and makes sure that it will never happen that two nodes access the same volume simultaneously. That also drastically decreases the risk of getting into severe file system problems if STONITH fails to do its work properly! cLVM is part of the SLES11 HAE, so you can use it without any problems.

In the procedure that is described below, you'll learn how to create cLVM volumes. This procedure consists of two parts: you'll need to create some resources in the cluster first that support clustered logical volumes. Without configuring these resources, you may have locking issues accessing the volumes. After doing that, you need to create and configure the LVM volumes themselves.
To prepare the cluster to handle cLVM volumes, there are two components you need to configure. First, you need to create a resource for the Distributed Lock

Manager (DLM). This component ensures that if one node has access to a volume, all other nodes know about it. Next, you need a supporting resource for cLVM. This has nothing to do with specific logical volumes, it's just a resource that makes sure that all supporting modules are loaded in the cluster before you start working on the cLVM's themselves. Since the resources take care of the preliminary conditions to load cLCM, you can create them as clone resources to be loaded on all the nodes. The following procedure describes how to do this.

1. Before creating the resources themselves, you need to tell the LVM stack on the host machines that the cluster will take care of locking. To do this, you need to change a setting in the /etc/lvm/lvm.conf file. In this file, by default you'll find the option locking_type = 1, you should change this to locking_type = 3.

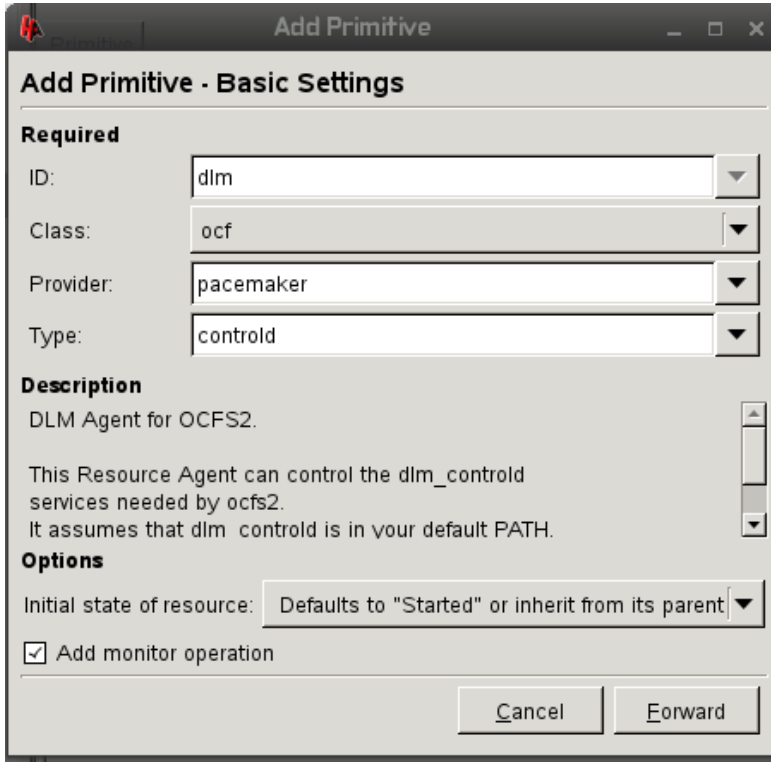
2. After changing the lvm.conf on all nodes in the cluster, you can start the hb_gui interface. In there, select Configuration > Resources and click Add. This opens the interface from which you can add new resources. As you want to create a resource that is to be started on multiple nodes simultaneously, you need to select the Clone resource type. Give the resource the name dlm-clone and make sure that the initial state of the resource is set to Started. Next click Forward to go on to the next step.



Create a clone resource with the name dlm-clone and make sure it is started automatically.

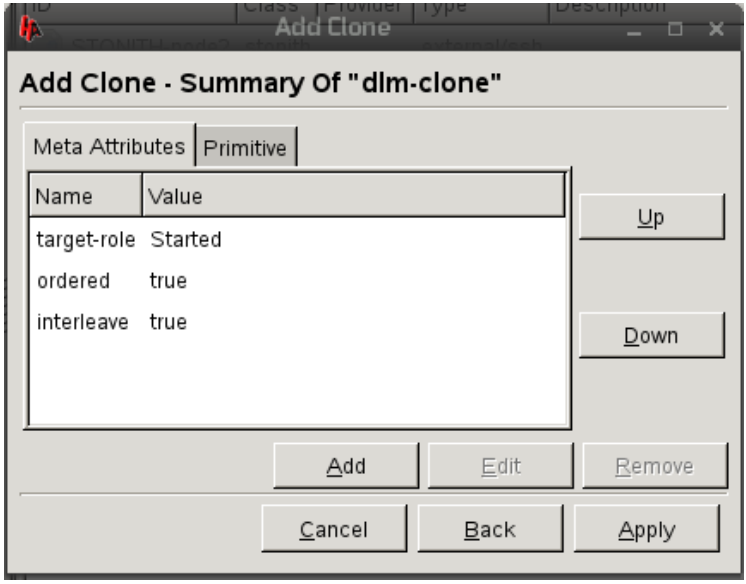
3. Now the utility asks you what sub-resource you want to create. This is the actual process that is going to be loaded on all of the individual nodes. Select Primitive and then click OK to continue.

4. Now give the ID “dlm” to the resource and select the class ocf, provider pacemaker and type controld. After selecting these, you can click Forward to proceed.



Configuring the DLM clone sub resource.

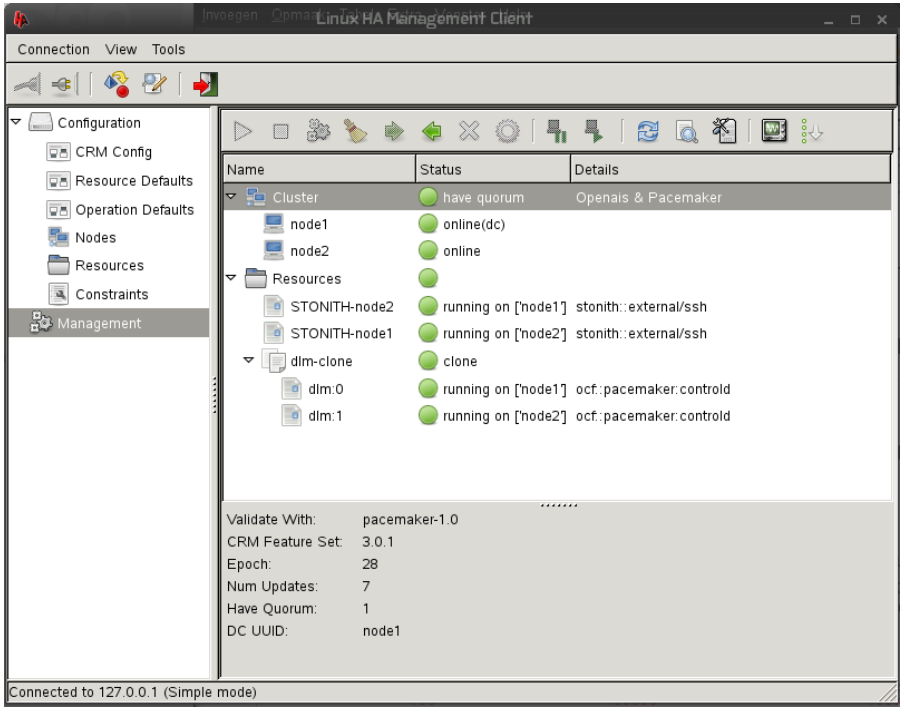
5. Now click Apply. This brings you back to the window where you specify the attributes of the clone itself. In here, select the Meta Attributes tab and click Add. Now add the attributes Ordered and Interleaved and make sure that both have the value true. The attribute ordered makes sure that the instances of the clone resource are started after one another. The option Interleaved is applied if there is a colocation constraint between two clone resources. This option makes sure that the same instances of the clone resource will always stay together. By using these two attributes, you will increase the stability of the cluster. Now click Apply to add the clone resource to the cluster.



Make sure that the clone resource is configured with the attributes Interleaved and Ordered, which both have the value True.

Tip: *All attributes that you can be used in the cluster, are described in the crm.dtd file. You can find this file on each cluster node in /usr/share/pacemaker. If something is not clear to you, it's a good idea to open this file to find the exact description of any given attribute.*

6. Now go to the Management part of the hb_gui interface. You should see the clone resources and you should see that all instances of the resource have been started on the cluster. Do not proceed if this is not the case!



The dlm resources need to be operational before you proceed.

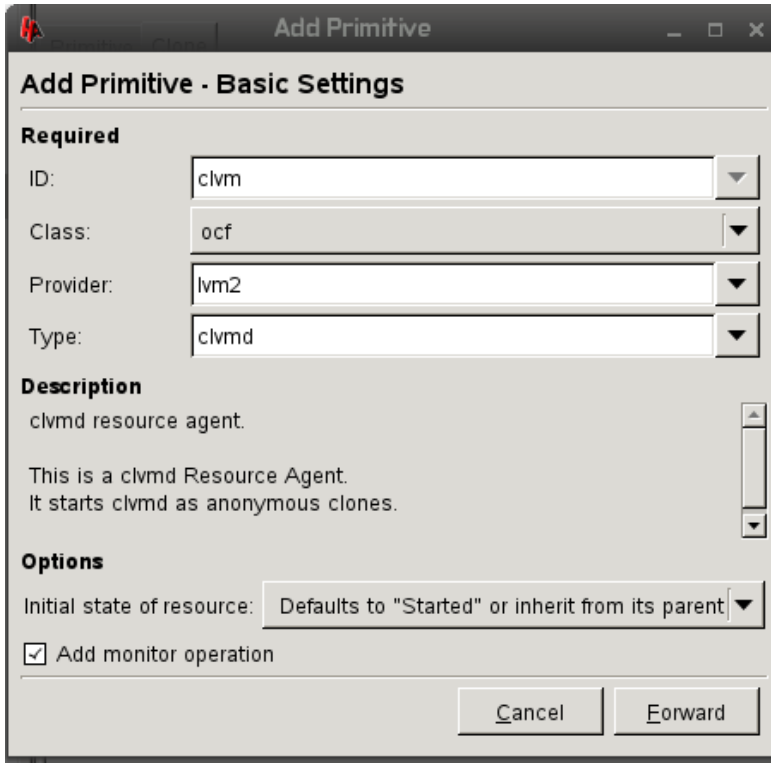
Tip: *In some situations you'll find that the DLM module cannot be loaded. The related error in /var/log/messages is as follows:*

Dec 23 12:30:20 nd1 lrmd [8192]: info: RA output: (dlm:1:start:stderr) FATAL: module '/lib/modules/2.6.27.39-0.3-xen/kernel/fs/dlm/dlm/ko' is unsupported Use --allow-unsupported or set allow_unsupported_modules to 1 in /etc/modprobe.d/unsupported-modules

The error message already gives the solution for this problem, open the file /etc/modprobe.d/unsupported-modules with an editor and make sure the parameter allow_unsupported gets the value 1. This should take care of the problem. However, this will mark your kernel as tainted, which may give you problems if you request support from third parties. Without this option however you cannot load the DLM, which prevents you from creating a Xen HA environment. And after all, you've paid for the SLES HAE, which gives you access to Novell support for this product.

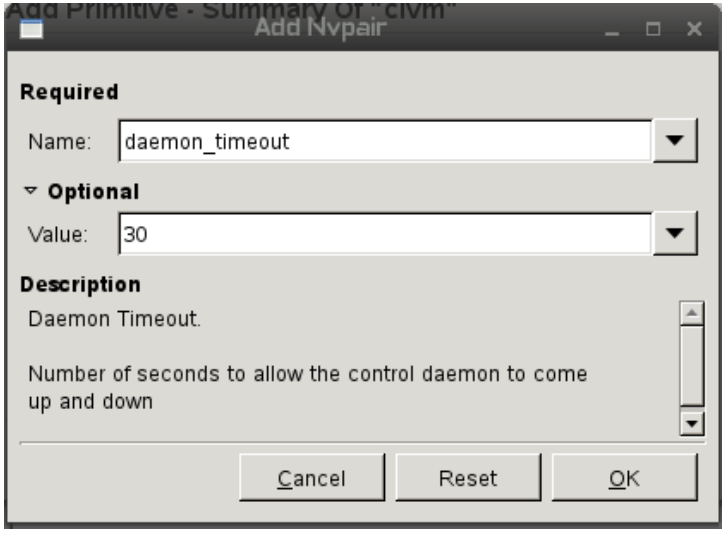
7. Now you can go on, configuring the cLVM resources. To make these changes, from the Linux HA Management Client, select Configuration > Resources > Add and indicate that you want to create a clone resource. Here you should set the Initial State of the resource to Started. Give the clone resource the name clvm-clone and next select Primitive as the sub resource type.

8. For the primitive, you can use the id clvm, the class ocf, the provider lvm2 and the type clvmd. The configuration window should now look as in the figure below. Verify that this indeed is the case, and then click Forward to proceed.



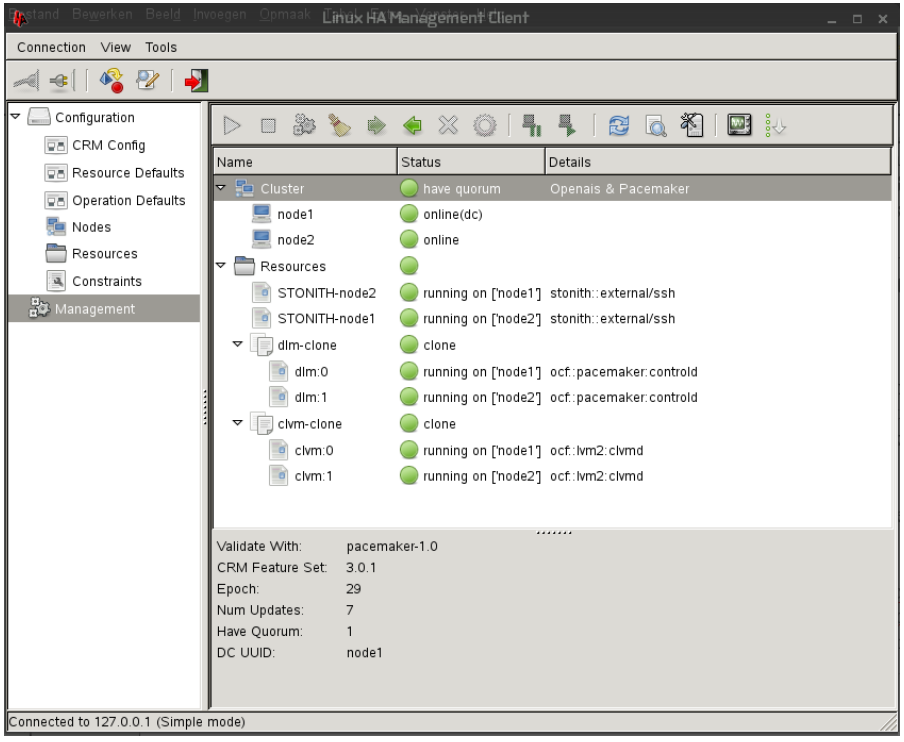
Creating the cLVM resource.

9. Now activate the Instance Attributes tab and on that tab, click Add to create a new attribute. This must be the attribute Daemon_timeout, which should have the value 30. With this, you tell the cluster that the cLVM daemon has 30 seconds to be started or to be stopped. After selecting this, click OK to save the setting and next click Apply to save the subresource attributes.



Configure the Daemon_timeout attribute to tell the cluster how long it should wait for the cLVM process to get activated.

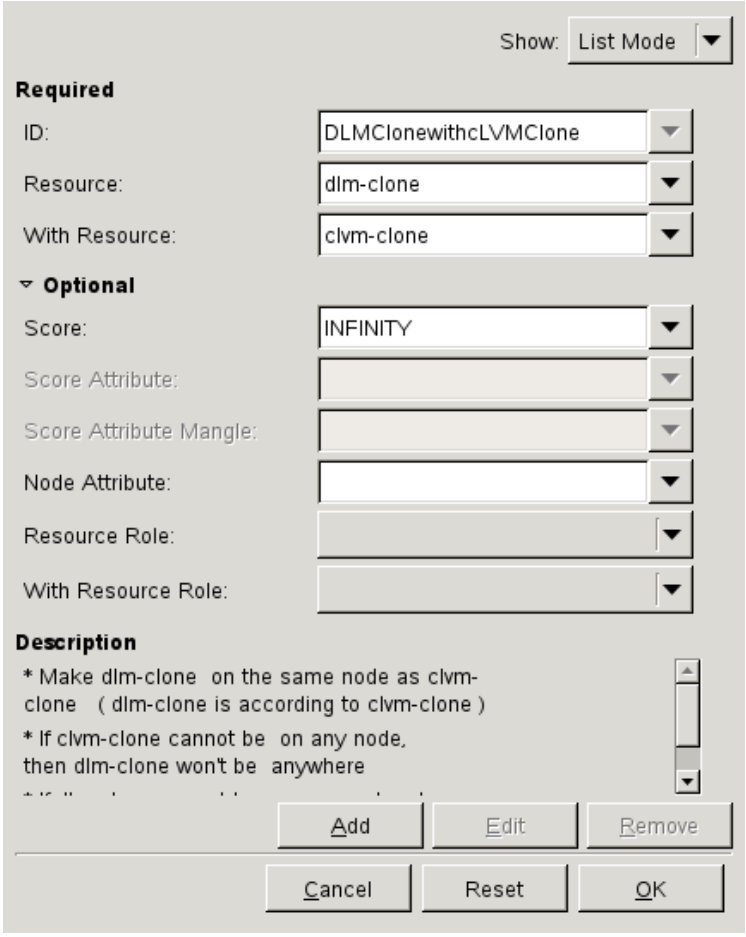
10. You are now back in the window with the properties of the clone itself. In here, click the Meta attributes tab. Also add the ordered and interleaved attributes for this resource, both should have true as their value. After doing that, click Apply to change the configuration. Next, have a look at the management part of the hb_gui and verify that all resources have been started properly (this can take about half a minute). Make sure that everything has been started before you proceed, you cannot configure the next resources if you have a problem at this point.



Make sure that the resources have been started before you proceed.

11. At this point, you have to add two constraints. First, there is an order constraint which is going to determine that the cLVM clone can be loaded only once the DLM clone has been loaded. You also need a colocation constraint that makes sure that DLM and cLVM are always started together on the same machine. To do this, in hb_gui click on Configuration > Constraints > Add and select the Resource colocation option.

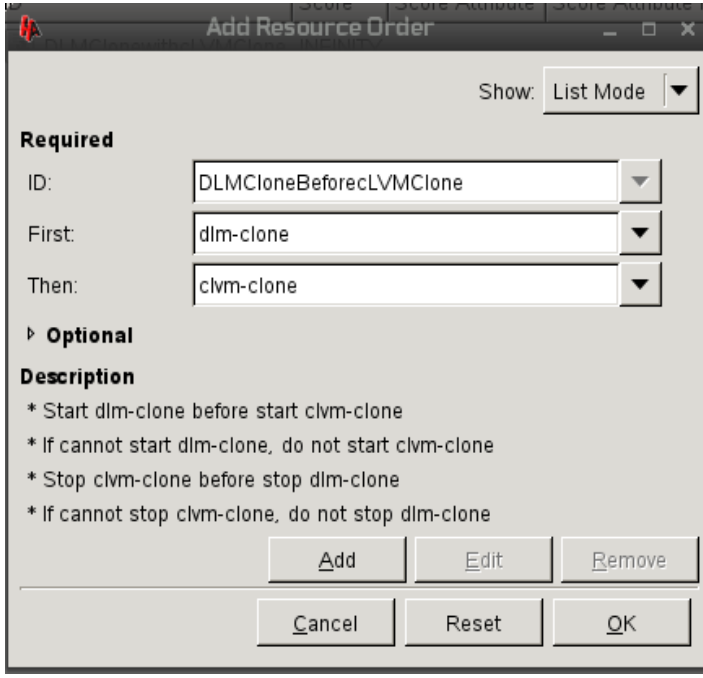
12. Now give an ID to the constraint, for instance DLMCloneWithcLVMClone. Next, from the drop-down list at the Resoruce option, select the dlm-clone resource and from the drop-down list at the With Resource option, select the clvm-clone resource that you have created before. At the end, give this constraint as score the value INFINITY, which means that they will always be started together. Next, click OK to save and apply



The colocation constraint makes sure that both clone resources are always started together.

13. Next, you also need to create a constraint that makes sure the DLM is started before the cLVM resource. To do this, in hb_gui select Configuration > Constraints > Add and next select the option Resource order.

14. Now give the constraint an ID, DLMCloneBeforecLVMClone would be a nice ID, as it shows exactly what the constraint is doing. At the First option, select the DLMClone resource, and then select the cLVMClone. Next click OK to save and apply the settings.



You need the order constraint to make sure that first the DLM clone is started and only after that, the cLVM clone.

15. You are now ready creating the supporting resources for clustered LVM. Verify thoroughly that everything works properly before you continue and resolve all open problems. It is better to build your house on stone than on sand.

5.1.1 Creating the Logical Volumes

Now that all preparation has been done, it's time to create the logical volumes themselves. Basically, that would mean that you have to do pvcreate, vgcreate and after that lvcreate to create the volumes. Following that, you need to create the appropriate resources in the cluster that take care of loading the logical volumes without any conflicts. Creating these logical volumes however is different from doing it in a stand alone environment, as the cluster is involved in taking care that the logical volumes are loaded safely. In theory it looks easy enough, in practice you'll have to do a special trick to make sure it all works.

The problem is that you cannot create a logical volume directly on a multipath device. It should be possible to do it, but in many situations it just doesn't work. So you'll first need to stop multipath, stop the cluster as well and then create the LVM logical volumes on the underlying devices. In the following procedure you can read how to do this.

As mentioned, you'll probably have a problem setting up LVM while multipath is operational. It might work though. So before you start the procedure described below, it's a good idea to try the quick solution. In this solution, you are going to tell the LVM configuration file lvm.conf that device mapper devices can be used as well. Do this by including the following line in the lvm.conf on all the servers:

```
types = [ "device-mapper", 16 ]
```

Next, verify that multipathd is active in the runlevels and reboot your server. After rebooting, use the pvs command to verify that the physical volumes have been created successfully. Now continue with step 5 of the procedure that is described below. If you've had an error creating this configuration, you can try the solution that is described in the procedure below.

1. Enter the multipath -l command. This command shows you which multipath devices there are and which local devices are connected to them. In the listing below, there are two multipath devices, /dev/dm-0 and /dev/dm-1. Local disks /dev/sdb and /dev/sdd are connected to the dm-0, local disks sdc and sdc are connected to the dm-1.

   Listing: Before you start, you should have a look at the current multipath configuration.

   ```
   node2:~ # multipath -l
   149455400000000000000000010000009c0500000f000000 dm-1 IET,VIRTUAL-DISK
   [size=10.0G][features=0][hwhandler=0][rw]
   \_ round-robin 0 [prio=-1][active]
    \_ 3:0:0:1 sdc 8:32  [active][undef]
   \_ round-robin 0 [prio=-1][enabled]
    \_ 4:0:0:1 sde 8:64  [active][undef]
   149455400000000000000000010000008f0500000f000000 dm-0 IET,VIRTUAL-DISK
   [size=10G][features=0][hwhandler=0][rw]
   \_ round-robin 0 [prio=-1][active]
    \_ 3:0:0:0 sdb 8:16  [active][undef]
   \_ round-robin 0 [prio=-1][enabled]
    \_ 4:0:0:0 sdd 8:48  [active][undef]
   ```

2. Now you need to stop multipath. On some configurations, this will work perfectly when entering the command /etc/init.d/multipathd stop. Verify if this works for your situation. If it doesn't, use insserv -r multipathd, followed by pkill -9 aisexec and reboot your servers.

3. Once the servers are up again, as root open a console on one of the servers. From here, you need to issue the pvcreate command to mark the SAN drives for use in multipath. So following the example from the above listing, that would be pvcreate /dev/sdb and pvcreate /dev/sdc in this example. You may see some error messages, but these can be safely ignored.

4. Now switch on multipath again, using insserv multipathd and reboot your servers to activate the changes.

5. Once the servers are back on-line, you can enter the pvs command to display the logical volumes that are currently available. You'll notice that the physical volumes aren't available on all servers, that's no problem and you are going to fix this problem later in this procedure. The cLVM resource in the cluster makes sure that only one node in the cluster can access the logical and physical volumes. You can see the result of the pvs command in the listing below.

   Listing: Use the pvs command to get an overview of the devices that currently are available for LVM.

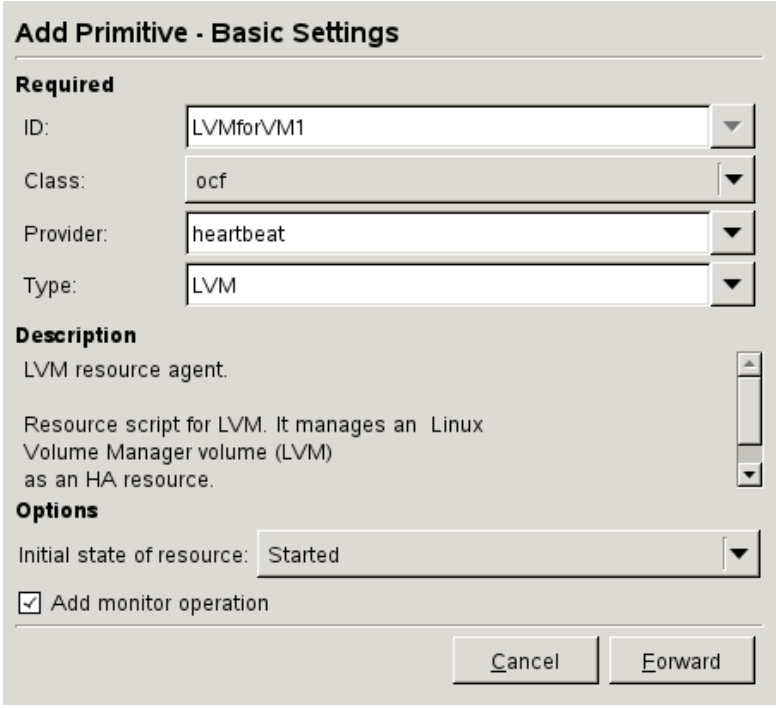
```
node1:~ # pvs
  PV         VG   Fmt  Attr PSize  PFree
  /dev/dm-0  vm1  lvm2 a-   10,00G 10,00G
  /dev/dm-1       lvm2 --    9,99G  9,99G
```

6. You can now continue on the server where pvs gave you a list of physical volumes as its result and create the LVM volume groups. The ultimate goal of this procedure is to create a logical volume which can be used as the storage back-end for the virtual machine. Each virtual machine needs one LVM volume. To create this LVM volume, you first need to create a logical volume, you need one logical volume per machine. To make it easier to connect all the pieces together at a later stadium, it's a good idea to use the name of the virtual machine for the volume group. Hence, you would create the volume group for the virtual machine vm1 using the command vgcreate vgvm1 /dev/dm-0.

7. As a last step, you need to create an LVM volume for each of the virtual machines. The LVM volume should be as big as the entire volume group. To create this volume with an example size of 10GB, you would use lvcreate -L 10G -n lvvm1 /dev/vgvm1. Repeat this procedure until you have created all the logical volumes you need.

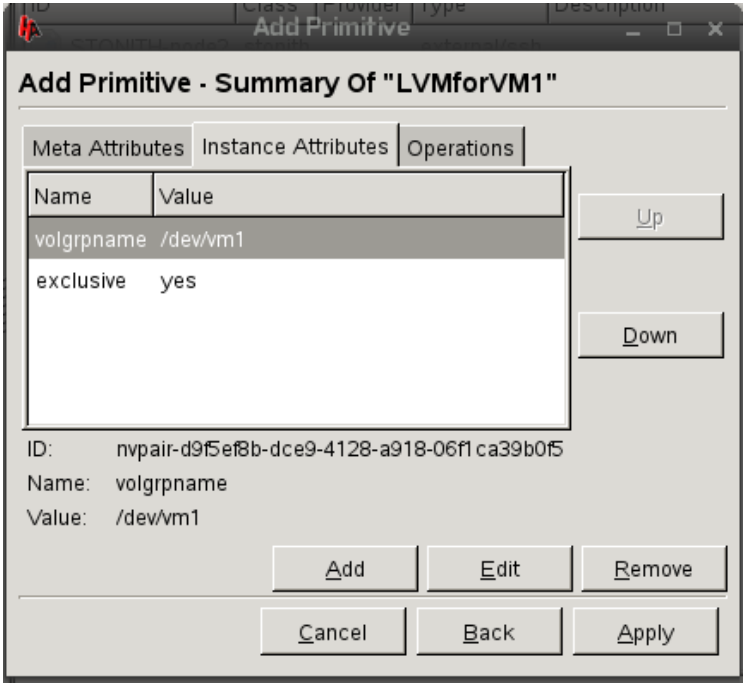
8. Now that you have created the logical volumes at the file system level, you need to create the resources in the cluster that determine where and how the volumes can be loaded. Before starting this procedure, you need to determine how you want to access the cluster aware logical volumes. There are two options: you can access them exclusively which means that only one node can access them at the same time, or you can configure them as volumes with shared access, where multiple nodes can access them at the same time. The latter definitely is a very bad idea, so you should make the volumes private volumes.

9. In hb_gui, select Configuration > Resources > Add and select the resource type Primitive. As ID for the volume, use a name that makes it easy for you to recognize what the resource is used for, such as LVMforVM1. Next select the class ocf, the provider Heartbeat and the type LVM. Give the resource the initial state Started and click Forward.



Creating a cluster resource for LVM

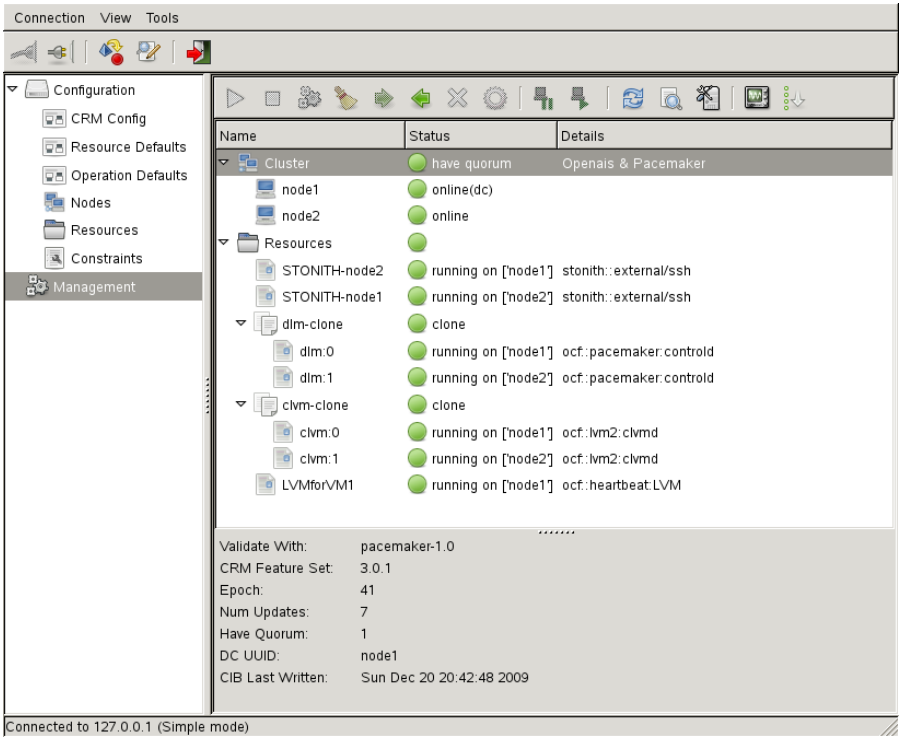
10. Now you need to enter two atrtibutes for the LVM volume. To start with, you'll have to specify the mandatory attribute volgrpname, which should refer to the device name of the LVM volume group. In this example, that would be /dev/vgvm1. Next, make sure that the parameter Exclusive has the value yes. The click OK to safe the attributes.



Specifying attributes to mount the LVM resource.

11. Now click Apply to add the LVM resource to the cluster. Check the management part of hb_gui and verify that the resource is indeed started.

Tip: *You will find in some situations that the resource is not started automatically and that there is no specific reason for that. If that happens, it can help to perform a resource cleanup. To do this, you have to activate the Management part in the Linux HA Management Client. Next, right-click the resource that isn't started and select Cleanup Resource. Wait a minute or so and you will hopefully find that the resource is started again.*



Before you continue, verify that the volume group resource has been started succcessfully.

12. You now need to create two constraints. First, you need a colocation constraint that makes sure that the cLVM resource and the LVM resource you've just created are always loaded together. You also need an order constraint that determines that the cLVM resource is loaded before the LVM resource. In former chapters you've learned how to create these constraints.

At this point you are ready with the configuration of the LVM volumes in the cluster. In the next chapter you'll learn how to create the Xen virtual machines themselves.

6 Setting up Shared Storage with OCFS2

In the days of SLES 10, it was common to set up OCFS2 as shared storage. In SLES 11 you can still do that, but the approach is a little different. This chapter describes how to set up an OCFS2-based shared storage. The information in this chapter is provided as an alternative to the information in the previous chapter, if you've already configured cLVM volumes, you can skip this chapter.

6.1 The big picture

When setting up OCFS2 as the storage solution, the overall process is going to be different. To make sure that you're going to apply the right procedure, you can find an overview of the steps you have to perform below:

1. Configure two LUN' s on the SAN, one generic purpose LUN on which you are going to store all the files that are going to be used as a storage back-end for the virtual machines and one small LUN that is going to be used to store the configuration files.

2. Set up the base cluster, including the configuration of STONITH as described earlier in this book.

3. Configure OCFS2 and create an order constraint to make sure that STONITH is loaded before OCFS2. In this chapter you can read how to perform this procedure.

4. Create the virtual machines. Instead of configuring LVM volumes as the storage back-end you would use the OCFS2 LUN as the storage back-end for the virtual disk files used by the virtual machines.

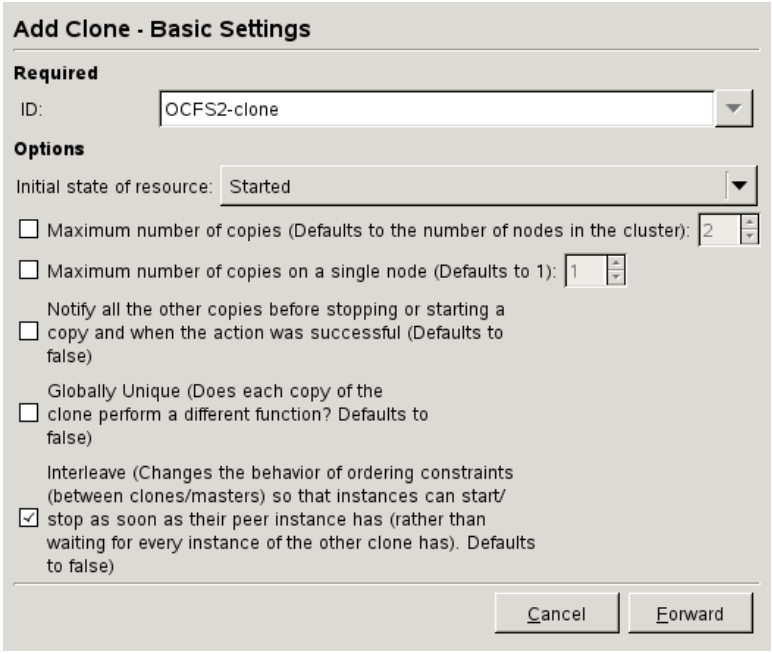
6.2 Setting up OCFS2

To apply the procedure described in this chapter, I'll assume that you've installed the OCFS2 packages. You can install them using YaST, from the SLES HAE repository as described earlier in this book. After installing the software, make sure to apply the following steps:

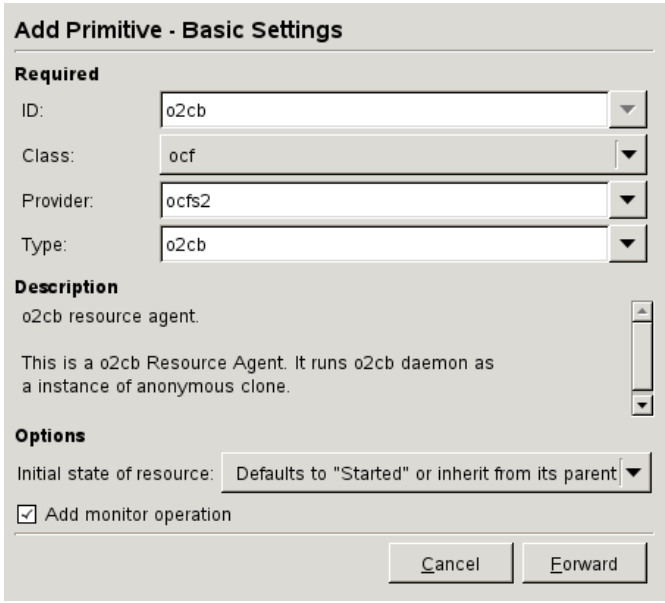
1. Create a clone resource for DLM. The procedure how to do this, is described in chapter 5.

2. From the Linux HA Management Client, select Configuration > Resources > Add.

3. From the Add window, select the Clone resource type and give it the id OCFS2-clone. Make sure that the resource also has the interleave option set to true (see figure below)



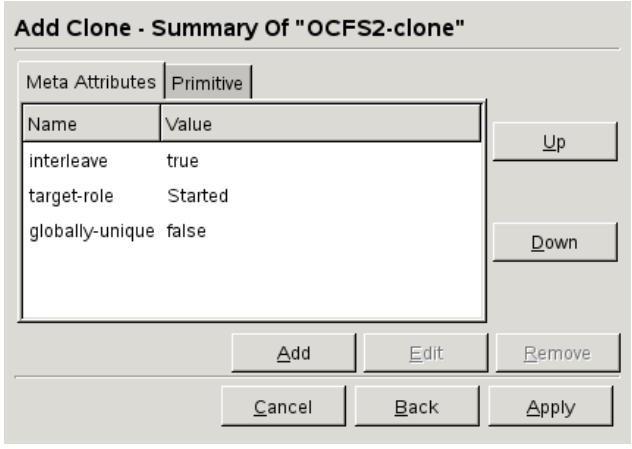
4. When asked what sub-resource you want to create, select Primitive and click OK. Enter the ID o2cb (which is the name of the service this resource is going to start), then select the Class ocf, the Provider ocfs2 and the Type o2cb. Make sure that the Add monitor operation is selected and then click Forward to proceed.



Creating the o2cb resource

5. From the Add Primitive widow, select the Operations tab. This shows a monitor operation which by default is set to 10 seconds. Select this property, click Edit and change the monitor interval to 120 seconds. Also make sure to set the timeout to 150 seconds.

6. Now click Apply to return to the Add Clone window. From this window, click Add and make sure that the parameter globally-unique is selected with the value false. The meta attributes tab for the clone resource should now look as in the figure below:



Meta attributes for the o2cb Clone Resource

7. At this point you need to set an order constraint that ensures that the o2cb service is started only after the dlm service has been loaded successfully. To do this, from the Linux HA Management Client, select Configuration > Constraints and then click Add.

8. Now select Resource Order and click OK. Then enter the ID dlm-before-o2cb, from the drop-down list at the option. First, select dlm, and from the drop-down list at the option Then, select o2cb. Next, click OK to safe your settings.

At this point you are done configuring the o2cb resource in the cluster. From the Management part of the Linux HA Management Client, you should now see that all instances of the clone resource are up and running.

6.3 Creating the OCFS2 Volume

Just creating a resource for OCFS2 in the cluster is not enough. You now have to create an OCFS2 volume as well. To do this, you perform the following tasks from one of the nodes in the cluster:

1. Use multipath -l to find the name of the device that you want to create the OCFS2 file system on. Assuming that you are using multipath, you must use one of the multipath device names, such as dm-0. (dm-2 in this example).

2. Consider the amount of nodes that you want to give access to the multipath device. In the mkfs.ocfs2 command, you are referring to the amount of nodes by using the -N command. For instance, use mkfs.ocfs2 -N 4 /dev/dm-2 to create a shared file system on /dev/dm-2 with access for four simultaneous nodes. The listing below shows what the result of this command looks like.

   Listing: Creating an OCFS2 volume with access for four nodes

```
node1:~ # mkfs.ocfs2 -N 4 /dev/dm-2
mkfs.ocfs2 1.4.1
Cluster stack: pcmk
Cluster name: pacemaker
NOTE: Selecting extended slot map for userspace cluster stack
Filesystem label=
Block size=4096 (bits=12)
Cluster size=4096 (bits=12)
Volume size=64436809728 (15731643 clusters) (15731643 blocks)
488 cluster groups (tail covers 22971 clusters, rest cover 32256 clusters)
Journal size=268435456
Initial number of node slots: 4
Creating bitmaps: done
Initializing superblock: done
Writing system files: done
Writing superblock: done
Writing backup superblock: 3 block(s)
Formatting Journals: done
Formatting slot map: done
Writing lost+found: done
Formatting quota files: done
mkfs.ocfs2 successful
```

3. Before you return to the cluster configuration, you need to confirm that the OCFS2 file system can mount on all nodes involved. To test this, you can perform a manual mount from all of the nodes that have the o2cb resource running on them. I like mounting on a temporary directory, just to confirm that it's working. To do this, use the following command:

   ```
   mount -t ocfs2 /dev/dm-2 /mnt
   ```

4. Don't forget to unmount all the OCFS2 mounts. If you don't do this, you'll see that you can't get the OCFS2 volume to mount in the cluster. To unmount the volume on all the nodes, use the following command:

   ```
   umount /mnt
   ```

If this gives you a “Device is busy” message, you are probably in the /mnt directory. Get out and try again!

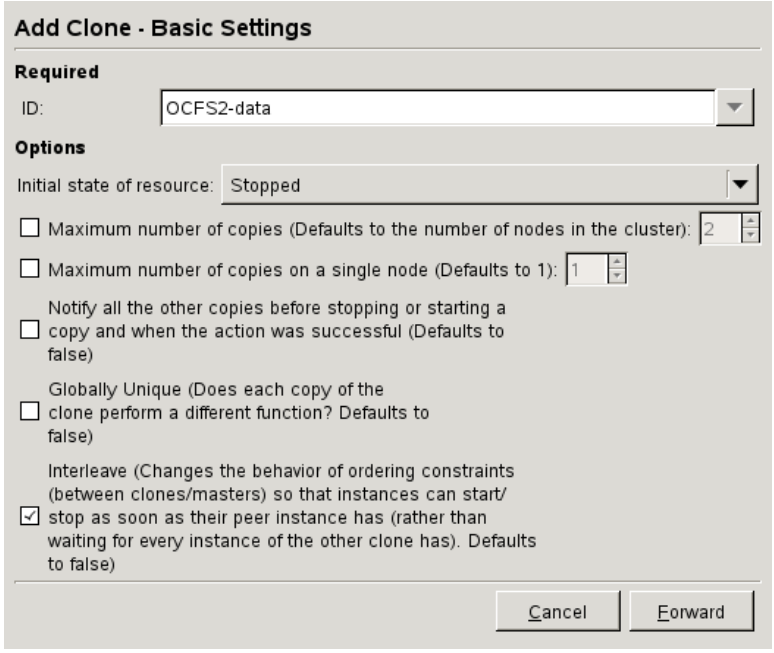
Now you need to repeat this procedure once more. For a Xen HA on OCFS2, you need two OCFS2 volumes, one to store the disk files on, which has to be big, and another one on which you are going to store the configuration files, for which 1 GB of available disk space is enough.

6.4 Creating Cluster Resources for OCFS2 Volumes

Now that you have set up OCFS2 successfully, you need to create a clone resource in the cluster, which takes care of mounting the volume on all the nodes. You have to take special care of the mount point for these volumes. The smaller volume must be mounted on /etc/xen/vm, the bigger volume must be mounted on /var/lib/xen/images to give access to the disk files of the virtual machines. The following procedure shows how to do this:

1. From the Linux HA Management Client, select Configuration > Resources > Clone and click Add. From the Add window, click Add. This brings you to the Add Clone window.

2. Give the clone the ID OCFS2-data and make sure the Interleave option is selected. Then click Forward to proceed.



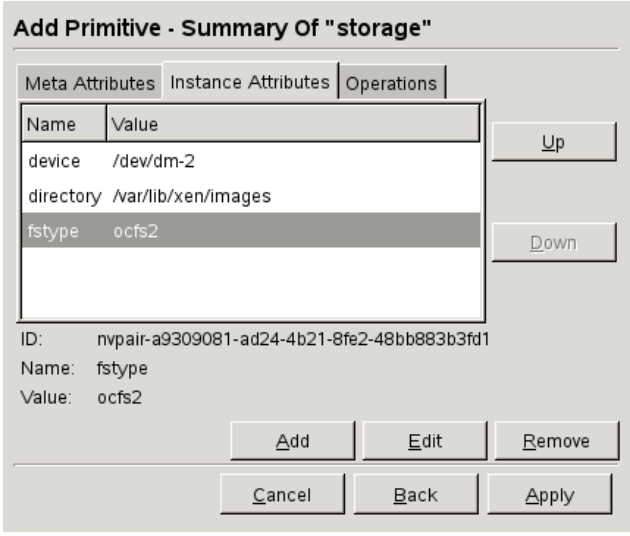
Creating a clone resource for the OCFS2 file system

3. From the Add window that appears, click Primitive and next select OK. This brings you to the Add Primitive - Basic Settings window. On this window, add the following information:

   - ID: storage
   - Class: ocf
   - Provider: heartbeat
   - Type: Filesystem

4. After clicking Forward, you are now at the summary window of the resource you've just created. On the Instance Attributes tab you need to specify all the attributes that are required to mount the OCFS2 volume:

   - device: /dev/dm-2 (replace by your specific device!)
   - directory: /var/lib/xen/images
   - fstype: ocfs2



Required parameters for an OCFS2 file system

5. Back in the Clone Summary window, activate the Meta Attributes tab and make sure the following attributes are set:

   - interleave: true
   - ordered: true
   - target-role: Started

Then click Apply to create the OCFS2 resource in your cluster. You can now repeat the same procedure to create a resource for the OCFS2 file system that is going to host the Xen configuration files in /etc/xen/vm.

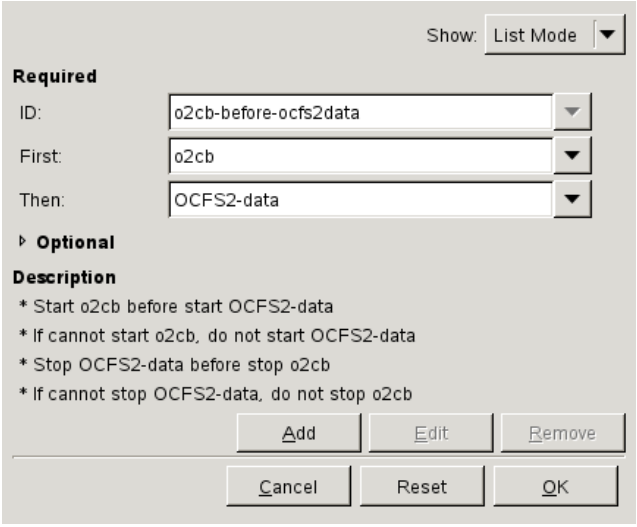
As the last part of the configuration, you now need to create an order constraint that makes sure that the OCFS2 file systems are only mounted if the o2cb resource has been started successfully. You need to apply this procedure for both of the file systems:

1. From Linux HA Management Client, select Constraints > Resource Order and click Add.

2. In the Add Resource Order window, enter the following parameters:

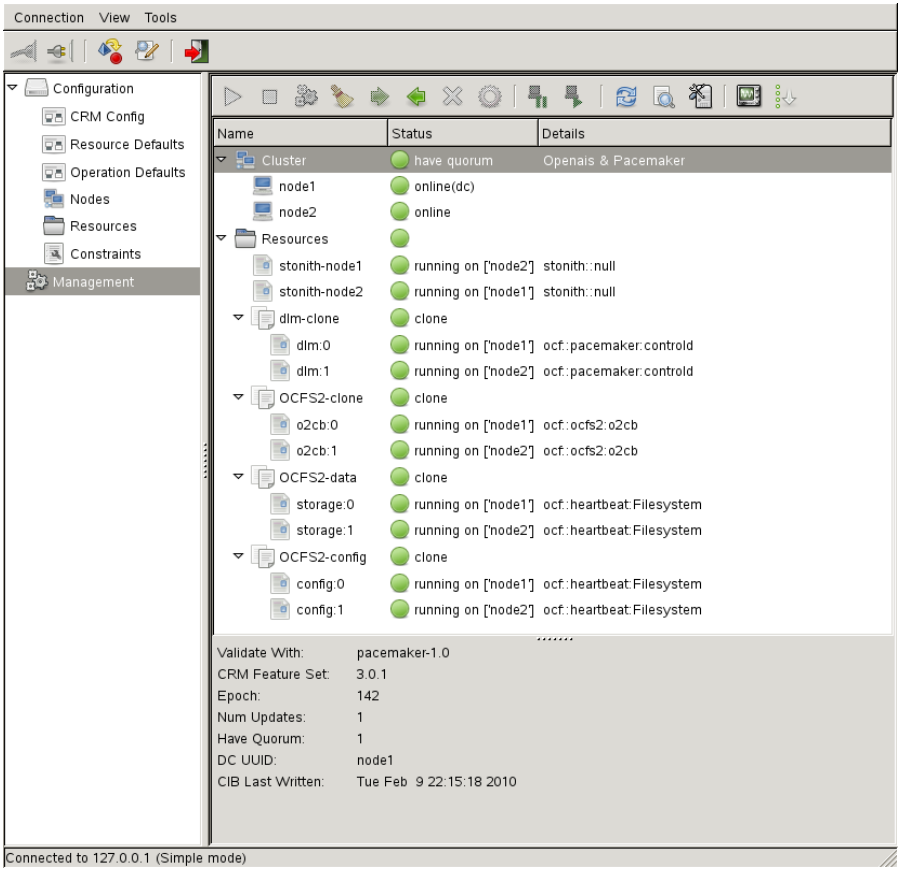
   ID: o2cb-before-ocfs2
   First: o2cb
   Then: OCFS2-data

   Repeat this procedure for the OCFS2-config volume as well.



Creating an order constraint to ensure the correct load order

You are now ready to configure the OCFS2 resources on the cluster. You should now verify that everything is up and running. If it is, you can go on to the next chapter, where you will create Xen virtual machines and resources to run them on the cluster.



When all looks good, it's time to move on to the next chapter

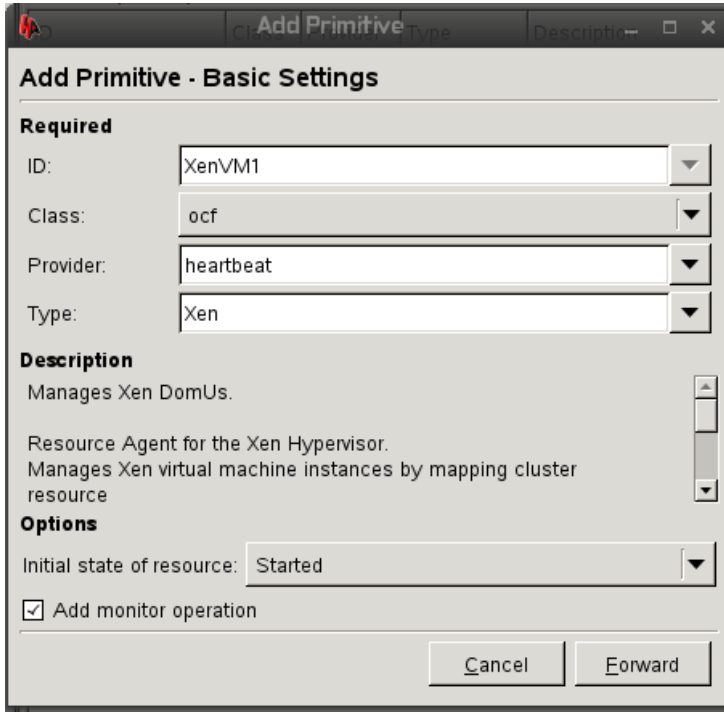
7 Setting up the Xen Environment

Now that you have the LVM volumes and resources operational, you can proceed with the next step: configuring the Xen virtual machines. This is a task that you should be able to do without too much trouble from the virt-manager utility, which is the default utility on SLES to manage virtual machines. There are some particularities for using this utility in a cluster environment though. The following procedure shows you what to do.

1. In hb_gui, find out on which host the LVM logical volume that you want to use as the storage back-end for the virtual machine is activated. On that host, use the virt-manager command to start the Virtual Machine Manager interface.

Tip: *You can of course also use the hb_gui to migrate the logical volume to the host where you need it. Right-click the volume resource and from the menu, select Migrate Resource. You'll now see a window where you can use the To Node option to indicate to which host you want to migrate the volume.*

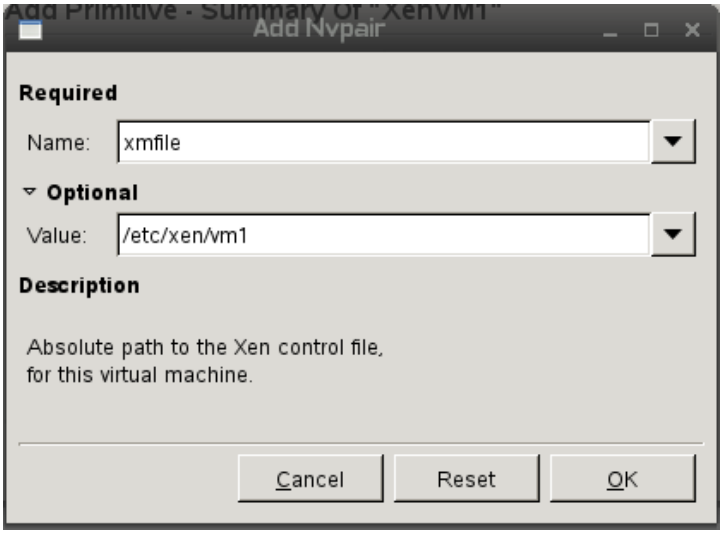
2. In Virtual Manager, double-click the localhost line. This changes the status of the local host to Active, which means that you can start creating virtual machines.



To start creating virtual machines, you need to activate the local host.

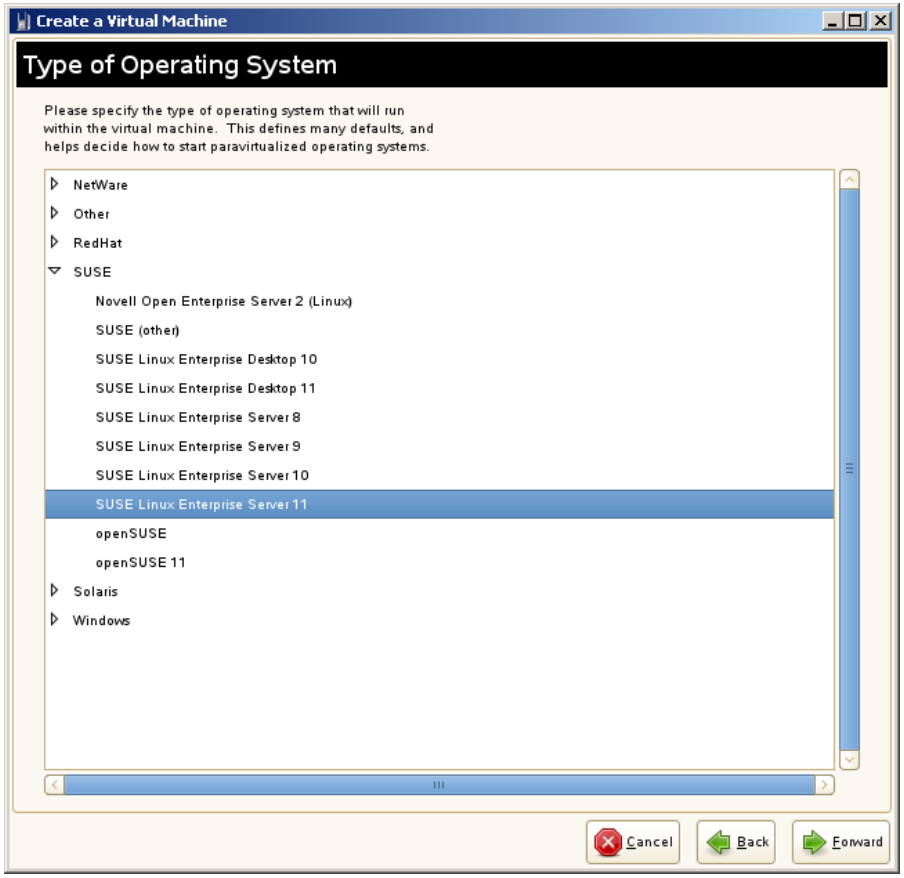
3. In the lower part of the Virtual Machine Manager interface, click New to create a new virtual machine. This starts the Create a Virtual Machine wizard. In the first window that is started by this wizard, click Forward to proceed.

4. The wizard now asks if you want to create a new virtual machine, or if you prefer using an existing disk that already has a complete virtual machine installed on it. From this window, select "I need to install an operating system" and click Forward to continue.



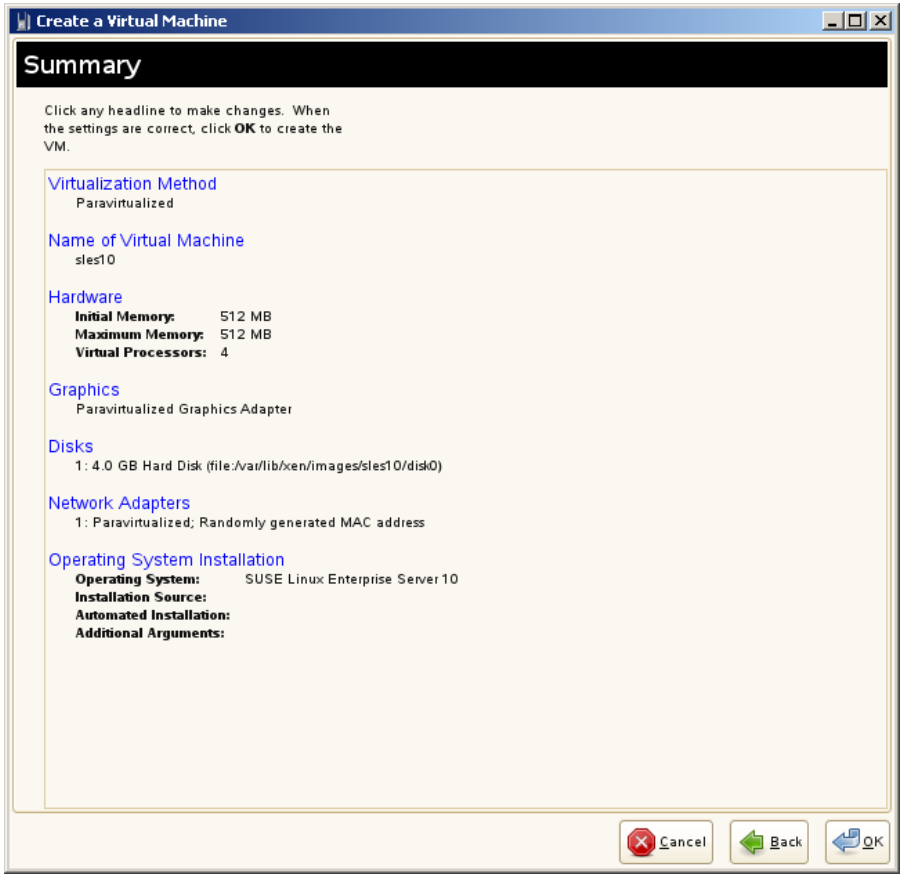
Select "I need to install an operating system" and click Forward.

5. Now specify which operating system you want to install. In this example I'll install SUSE Linux Enterprise Server 10, you can install any other operating system as well. The only requirement is that it uses the same processor architecture as the host operating system. That means that you cannot install 64-bit on a 32-bit host (the opposite is possible) and you cannot install SPARC software on an Intel platform.



In Xen, you can install the operating system of your choice

6. You now see the Summary screen, from which you have to add the further configuration of your virtual machine. To start with, you need to specify the virtualization method you want to use. If your operating system can be installed in paravirtual mode (which is the case if the software offers the paravirtualization option), you should always install in paravirtual mode, as it offers a much better performance. In all other cases, you need to select Full virtualization.

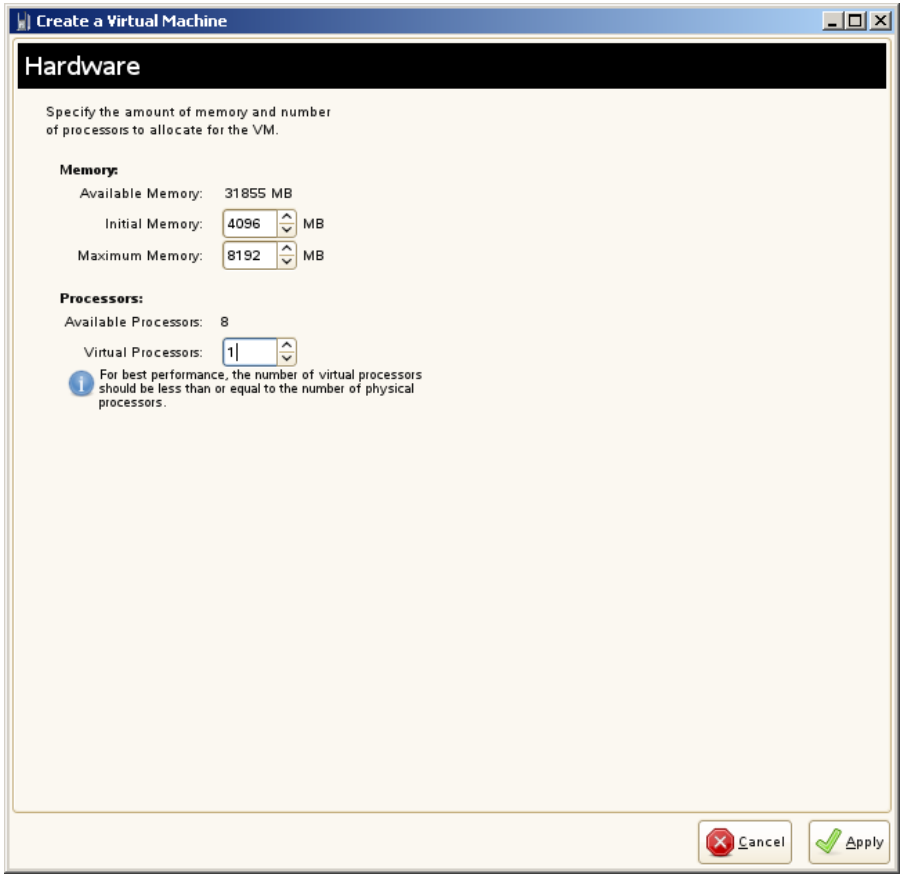


If possible, install the operating system in paravirtual mode

7. Next, you need to give a name to the virtual machine, after which you click the Hardware link to specify how much RAM and CPU cores the virtual machine is allowed to use. Hopefully, you have already made a calculation for the amount of RAM your machine needs. Set the result of this calculation as the Initial Memory, for instance, set it to 4096 to give your virtual machine 4GB of RAM. It's also a good idea to specify the Maximum Memory to a considerably higher amount of RAM, Xen offers the possibility to increase the amount of available RAM without restarting the virtual machine. As you can imagine, this is very useful if after some time you find out that the 4GB wasn't enough initially. Setting the maximum amount of memory to 8GB doesn't mean that your server is going to use all of it immediately anyway.

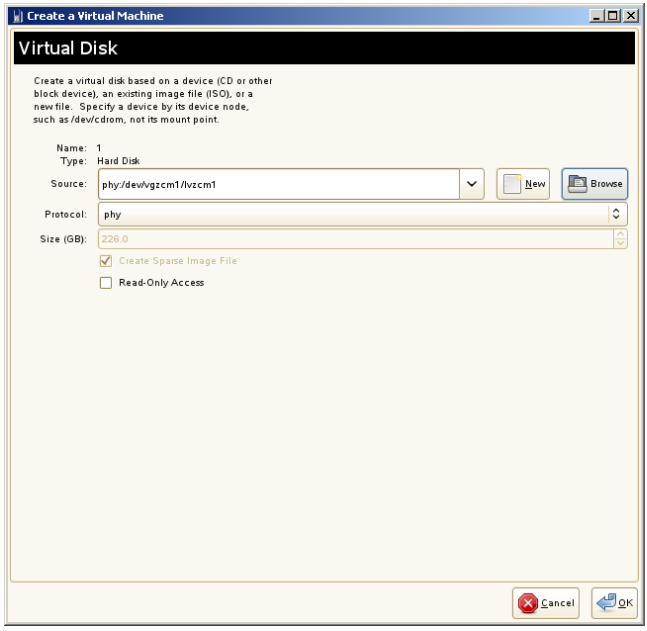
Apart from the RAM requirements, you also need to indicate how many CPU cores the virtual machine is allowed to use. Make sure that you never select more than the actual amount of cores that is physically available in your server! If you don't

expect the virtual machine to have a specific benefit from using more than one CPU core, it's even better to limit the amount of virtual CPU's to one. This prevents you from having to optimize your server for a multi CPU environment.



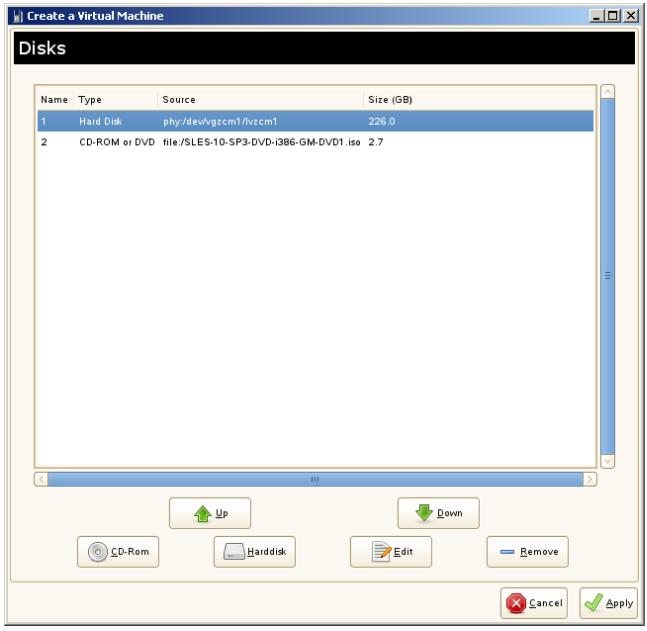
Using the Maximum Memory parameter you can make it easier to allocate more memory to a VM later.

8. At the Disks option, you have to indicate what the virtual machine can use as its hard disk. You probably have guessed it already, you'll use the LVM volume that you have created in the cluster for this purpose. After clicking the link Disks, you'll notice that the virtual machine already has obtained a default disk. This is a 4GB disk file which you can safely delete. Next click New and browse to the location where the device you want to use exists. You should look for a directory with the name of the volume group, that exists in the /dev directory. In this directory, you'll find the device file that you need to address the logical volume you've created before.



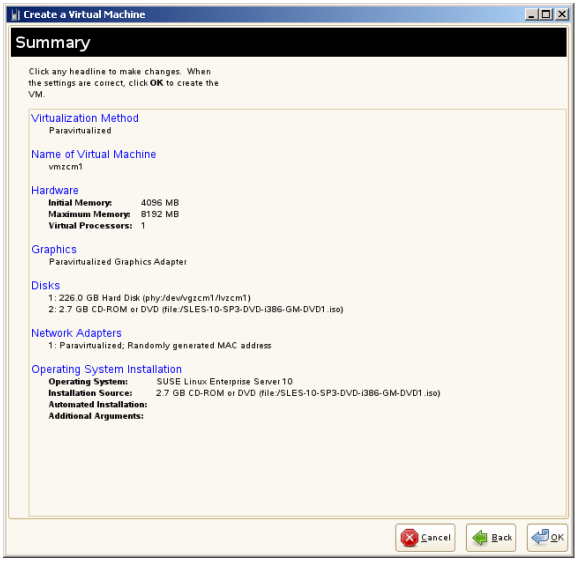
Use the LVM volume that you've created before as the storage back-end for the hard disk in the virtual machine.

9. Back in the Disks window, it's also useful to indicate how you are planning to install the virtual machine. Assuming that you're using an installation-CD, you can click the CD-Rom button and browse to the device that you're using to address the optical drive (such as /dev/cdrom), or the location of the ISO file that you want to use for this purpose. Back in the Disks window, you should now have a hard disk as well as an optical disk. If this indeed is the case, click Apply to safe your changes and get back to the main menu.



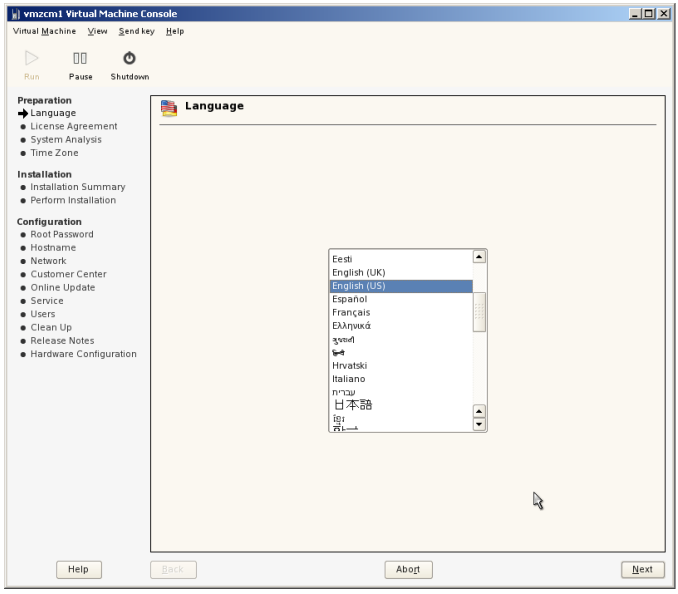
You have to add a hard disk, as well as an optical drive.

10. In case you have configured load balancing on the network, by using several bridges, you should now indicate which network adapter in the virtual machine you want to connect to which bridge. If you have one bridge only (or have no clue what this is about), you're ready at this point and can click OK now. This will start the installation of the virtual machine. Before you do so, it is however important to doublecheck that everything really is configured the way you want it to.



Check that everything is configured the way you want it to, before clicking OK to continue.

11. You'll now see that it takes a while for Xen to perform all preliminary steps to start the installation. This can take about thirty seconds. After that, a window is opened in which you can complete the installation of the virtual operating system. Perform this installation the way you have planned it.



At this point you can perform the installation of the virtual operating system.

***Note**: While installing a Xen virtual machine, you will see that you have two mouse cursors instead of one. This is annoying, but there's not much that you can do about it - it is a known problem to the VNC server that is used while doing the installation. Once the installation has been completed, there are other methodes to work on your virtual machines, and you won't have this problem no more.*

12. After installing the virtual machines, there is one last but very important step to perform. You need to copy the configuration of the virtual machine to all the other hosts in the network. You could do this by using an automated solution for file sharing, such as NFS shares or an OCFS2 volume, which can be reached by multiple nodes simultaneously, but you'll find out that that is too much work for something that can be accomplished easily by hand. You'll probably create new virtual machines occasionally only, so you might as well copy the configuration files to the other hosts manually. So at this point, make sure that you copy all the files in /etc/xen/vm to all hosts in the cluster and you'll be ready to proceed to the next step.

***Tip:** For some vague reason, you might loose the VNC connection from the Virtual Machine Manager and get a message that VNC is not able to connect with the hypervisor. There's an easy solution to this problem: just close the installation window that is failing to connect and get back to the Virtual Machine Manager. Next, select the virtual machine you were installing and click Open to get access to the console again.*

13. After manually copying the virtual machine configuration files, you need to make sure that all the virtual machines are known to the Xen management databases on the other hosts. To do this, you first need to perform a manual migration of the LVM volume resource for the virtual machine in the hb_gui to the host where you need to install the virtual machine in the database. Next, you can use the command xm create vmname to add the virtual machine to the Xen database on that host. After doing that, you can use the xm console vmname command to log in to that host. You need to press Enter once after doing this. This allows you to log in to the host. Now shut down the virtual machine and repeat this procedure on all other hosts in the cluster.

***Tip:** If you want to install a 32-bit virtual SUSE server that has more than 3 GB of RAM, you might see that the virtual machine refuses to start up again after the installation. If this is the case, you'll need to perform a new installation of the virtual machine, in which you have to select the Xen-PAE kernel. Also make sure that this kernel is loaded from the GRUB boat loader. This will fix the problem.*

7.1.1 Preparing for Live Migration

Amongst the cool features of Xen, is Live Migration. This means that you can migrate one machine to another, without any down time. The user will notice nothing at all while you are live-migrating the machine! If you are using live migration in a cluster environment, this also gives you the benefit that when you

shut down a host, the virtual machines on that host automatically migrate to another host in the cluster before the shutdown actually takes place.
To use live migration, you'll need to take some precautionary measures on the host machines. You'll have to change a setting in the /etc/xen/xend-config.sxp file, which you will find on every Xen host. The following procedure described what to do:

1. With an editor, open the file /etc/xen/xend-config.sxp and find the following lines:

```
#(xend-relocation-server no)
#(xend-relocation-port 8002)
(xend-relocation-hosts-allow '^localhost$ ^localhost\\.localdomain$')
```

2. Change these lines to look as in the following example:

```
(xend-relocation-server yes)
(xend-relocation-port 8002)
(xend-relocation-hosts-allow ' ')
```

   You should be aware that the xend-relocation-hosts-allow line from the example above will allow all hosts in the network to migrate virtual machines to this host. If you don't want that, you should apply some additional filtering using regular expressions. In the xend-config.sxp file you'll find some examples of regular expressions that you can use. Also make sure that the above lines occur once only in the configuration file, to avoid conflicts on your configuration.
3. Repeat the procedure described above on all host servers where you want to be able to perform live migration

4. To test if this procedure really works, you can now try it out, using the xm migrate --live command. For instance, to migrate the virtual machine vm1 to the host node2, you would use the following command:
   xm migrate --live vm1 node2

7.2 Creating Cluster Resources for the Virtual Machines

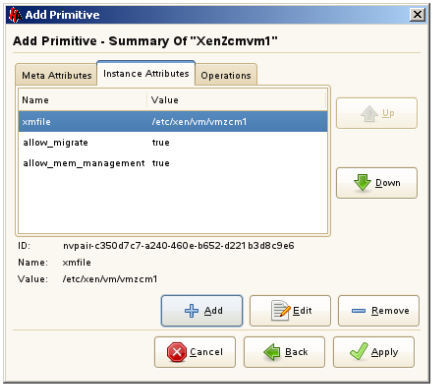
As the final step, you now need to configure the cluster to manage the virtual machine. This means that you have to create a resource for the virtual machine, and after that a colocation constraint that ensures that the virtual machine and the corresponding logical volume always stay together. From that follows the colocation with the DLM and cLVM resources, which in turn are required to start the logical volume successfully. Apart from these colocation constraints, you'll need some order constraints as well. After all, you only want to start the virtual machines after the logical volumes have been initialized succesfsully!
Before you start, make sure that you've stopped the virtual machines that you've just created. After creating the resource in the cluster, the cluster is going to start the virtual machine and that's not going to work if the virtual machine has already

been started somewhere else. Even worse: you may even risk severe file system damage if you try it anyway!

1. Now make a primitive resource, with the id of your choice (vmserver1, or anything else you like), make sure to select the class “ocf”, provider “heartbeat” and type “Xen”. (Please do notice that resources are not listed in alphabetical order, you'll first see all the resources of which the name starts with an uppercase and following that the resources of which the name starts with a lowercase. Make sure the initial status of the resource is set to started and next click Forward to proceed.

Creating a resource for the Xen virtual machine

2. In the Instance Attributes window, you'll see an attribute that has the name xmfile. This attribute needs the exact name and location of the configuration file that is used to start the virtual machine. For instance, it could be /etc/xen/vm/vm1, depending on the configuration that you are using. If you want to use live migration as well, you have to use the parameter allow_migrate and give it the value “true”. This option ensures that if a node in the cluster goes down, the cluster performs a live migration on the virtual machines that were running on that node. Using this option means that it will take a little longer before a node really is down (after all, you first need to live-migrate all virtual machines off of it), but this option also offers the advantage that users will experience no down time at all. Another parameter that you may like, is allow_mem_management. As it's name suggests, using this parameter makes it possible to adjust memory dynamically at a later instance, which is useful if you find out that the remaining host in your cluster doesn't have enough memory to host all of the virtual machines. For now however, the Pacemaker software offers limited abilities to manage this parameter from the cluster.



From the cluster, you can perform a limited amount of management tasks on the virtual machines.

3. Before doing anything else, you should make sure that the virtual machine is running nowhere in the cluster. If it is running, you will have serious problems while performing the next step. So use xm list on all hosts to verify that the virtual machine you've just been working on, is not active anywhere. If it is, use xm destroy <vm-name> to “pull the power plug” out of the virtual machine.

4. At this point you can save your changes. While doing that, the cluster will automatically take care that the Xen virtual machine is started.

As the last part of the configuration, you now have to create constraints. While creating the cluster, you ensured that the logical volume required by the virtual machine was available before starting the virtual machine. When the cluster tries to start the virtual machines automatically, you need to make sure that the volume is always loaded on the same host as the virtual machine, and that it is loaded before the actual virtual machine itself is started. So you need a colocation constraint, as well as an order constraint to take care of this. Once this has been done, you have your high availability cluster up and running.

8 Managing The Cluster

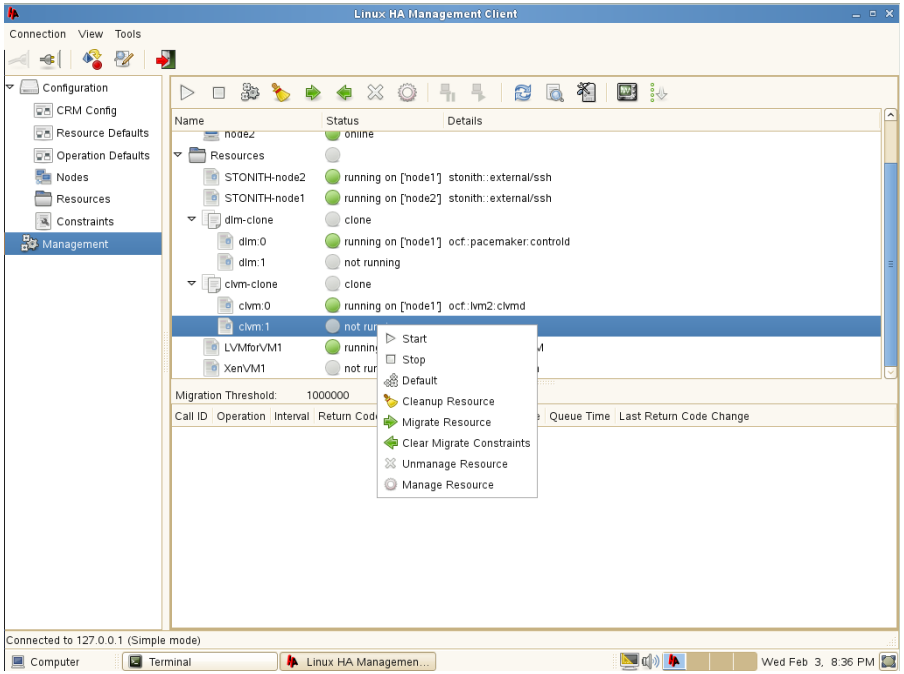
Now that the cluster is up and running, the real work begins. The advantage of a well-designed cluster, is that it doesn't involve that much work. The cluster basically runs itself and only in rare situations will it be required to perform specific management tasks. The purpose of this short chapter, is to make you familiar with the most important management tasks.

8.1 Cleaning Up Resources

On some occasions, you'll have to stop a resource and start it again. It may even happen that you have to delete a resource that is behaving the way you expected it to, and create it again as a new resource. In such situations, you may find that the resource doesn't disappear properly from the overview of resources in the hb_gui.

The reason why resources sometimes keep roaming, is that the cluster maintains the actual status of the resources and tries to keep it synchronized. If you're interested, you can display this status information using the cibadmin -Q command. This command will dump a lengthy xml configuration, containing the current status of the cluster. This status comes from the cib.xml, which is the actual state of the cluster that is synchronized between nodes in the cluster (and which you should never ever touch yourself!). In the file, you'll find a configuration part, which contains the resources that you've created, and the status part, which contains the actual status. In some cases however, the actual cluster status doesn't get synchronized succesfully. That may mean that after deleting a resource from the hb_gui, you might still see it roaming around, or that changes you've applied to resources don't get applied to the cluster. If this happens, it may help to clean up the resource. Cleaning up a resource forces the resource to be changed across the nodes in the cluster and may help in getting the proper information to other nodes.

To clean up a resource, just right-click the resource you want to update the status of, and select Cleanup Resource from the menu. This will immediately start synchronization of the resource and may resolve the issues that you are having with it.



If resource problems are due to outdated status information in the CIB, you might resolve them by cleaning up the resource.

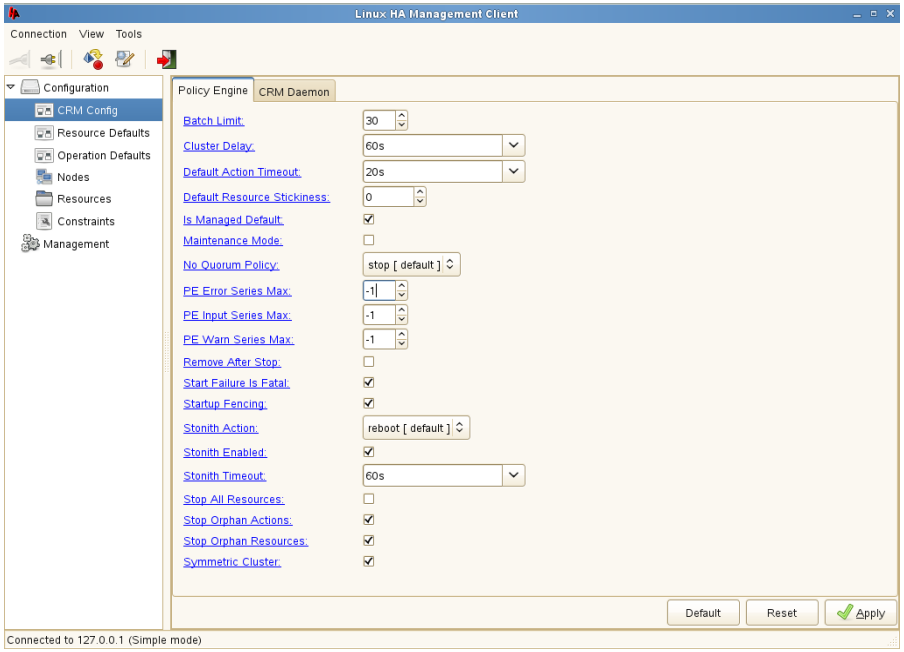
After selecting the Cleanup Resource option, the graphical application will prompt you on which node you want to clean it up. If you know for sure that the problems are related to one particular node not being updated properly, from the drop-down list, select that menu. Otherwise, just select All nodes to update status information on all nodes in the cluster.

8.2 Migrating Resources

Another common maintenance task, is the migration of resources. If a failure occurs on the cluster, resources will migrate away from the failing node automatically. The cluster automatically load balances migrated resources, which means that they are spread across the remaining nodes, thus trying to avoid one node in the cluster from being overloaded by migrated resources. The default behavior, is that when the node that the resource originally was loaded on, comes back in the cluster, the resource will fail back to the original node. If you don't like this, you can change the default resource stickiness. You can find this parameter on the properties of the CRM, the Cluster Resource Manager, which is the core component of the cluster.

If you don't like the behavior that resources automatically fail back to their original nodes, you can change the value of the Default Resource Stickiness. The following procedure describes how this works:

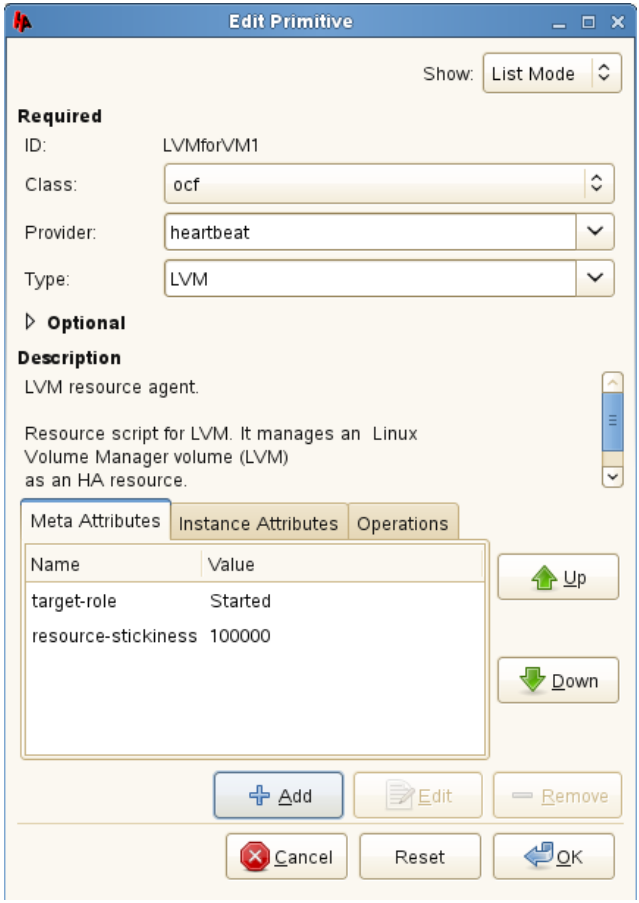
1. From the Linux HA Management Client, select Configuration > CRM Config. This gives you the current values that are used by the CRM.



On the CRM Config object you can set default behavior for the cluster.

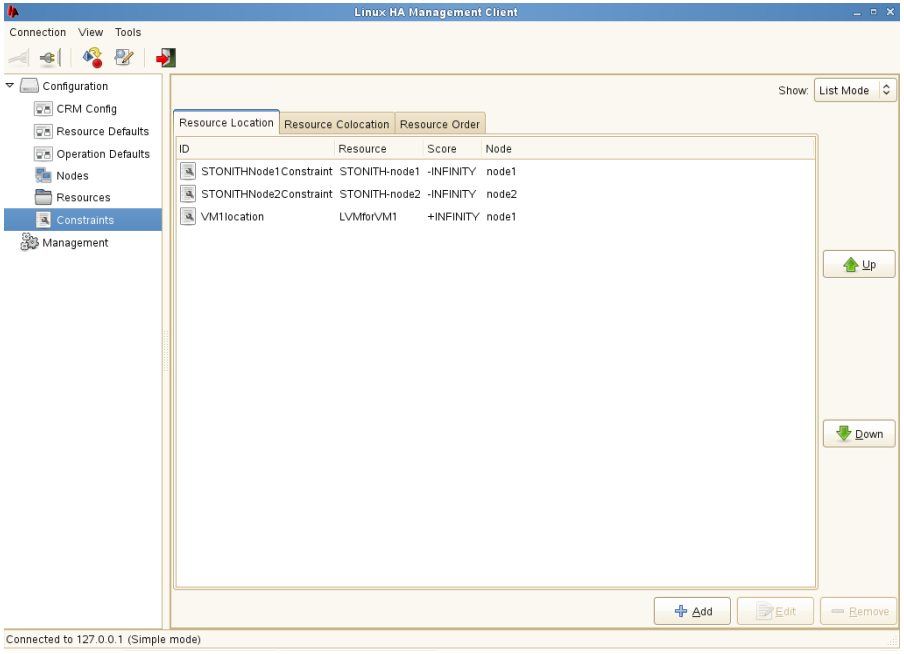
2. Select the Default Resource Stickiness parameter. The current value of 0 indicates that the resource will try to stay where it is. It also implies that the resource will move back to its original location after a critical issue in the cluster has been resolved. Change this value to 100000. That means that the resource will never move away from it's current location, unless the node that it currently is on goes down.

Instead of setting the resource stickiness cluster-wide, you can also use it on specific resources. To do that, select the resource from the cluster and click Edit to change its values. Next, select the Meta Attributes tab and click Add. From the drop-down list, you can now select the Resource Stickiness. Set it to the value that you'd prefer for your environment.



You can also specify resource stickiness on individual nodes.

When a resource is migrated after a node failure in the cluster, it will follow the resource stickiness that is set for the resource. If however you migrate a resource by hand, a location constraint is created for the resource automatically. This is something that can cause confusion, because the location constraint may cause the resource to fail to automatically migrate over to another node if a node goes down. Therefore, after a manual migration, you always have to clear the migration constraint. This is an easy action that you can perform from the menu that appears after right-clicking a resource. So after a manual migration, just right-click the resource and from the menu select Clear Migration Constraint. That will allow the resource to go anywhere it wants to in the cluster. In the figure below you can see a migration constraint that has been set to +INFINITY after the LVMforVM1 resource was manually migrate to node1.



After manually migrating a resource, a migration constraint is created for the resource.

8.3 Changing the Resource Management State

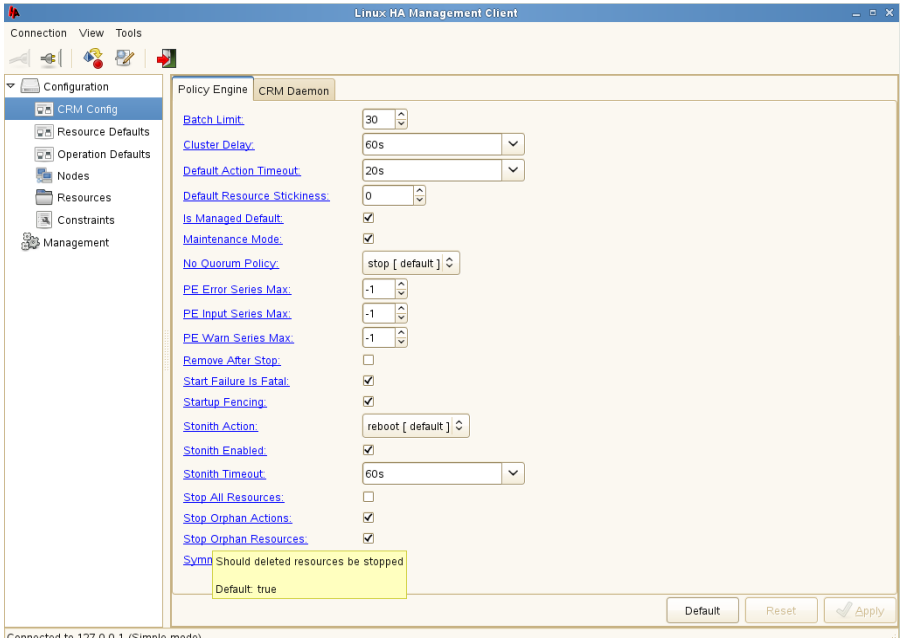
In some situations, such as an update of the cluster software, you may have to shut down the cluster software completely. That may be a problem though, when shutting down the cluster software, all resources that are managed by the cluster would go down also, which is not what you've created a high availability solution for. To prevent this, you can change the state of the cluster resource. By default, resources are in a managed state, which means that the cluster takes care of the resource and it cannot run without the cluster software being operational. If you put it in an unmanaged state, the resource will run all by itself.

To put a resource in unmanaged state (and to put it back in managed state), you can work on individual resources, or you can set the state for the entire cluster. To set it for the entire cluster, apply the procedure below:

1. From the Linux HA Management Client, select Configuration > CRM Config. This shows the current properties of the cluster.

2. Click the Check-box next to the Maintenance Mode option and click Apply to safe your changes.

3. Now have a look at the Management part of the hb_gui, this will show that all resources in the cluster are now in unmanaged mode.



After putting the cluster in maintenance mode, all resources reach the unmanaged state.

4. When you're doing maintenance on the cluster, go back to the properties of the CRM Config and remove the check-box next to the Maintenance option. This allows the cluster to take back control of the resources so that your resources will be safe again.

Like with so many options, you can apply them for the entire cluster, or for individual resources only. The options to manage Managed Mode for individual resources, are also in the menu that appears after right-clicking the resource. Select Unmanage Resource to put an individual resource in unmanaged mode, select Manage Resource to put it back in managed mode once you've finished your work with the cluster.

8.4 Verifying and Synchronizing the Cluster

In some situations, you'll see that it just doesn't work the way you want it to. There may be two reasons for that. First, you may find that the changes in the cluster don't get synchronized fast enough to the other nodes. Another issue may arise if there are errors in the configuration, which can even happen if you've only created resources from the Linux HA Management Client!

For both of these cases, there's a command that might help you. First, you can force immediate synchronization of the Cluster Information Base (CIB) between the nodes by using the cibadmin -S command. This will push the configuration to the other nodes and make sure they are up-to-date as well.

In case you suspect that there are errors in the current cluster configuration (or if a log message in /var/log/messages tells you that there are errors), you can run crm_verify -L. This command will give you a list of all currently existing errors, so that you can fix them.

9 Appendix: Creating an Open Source SAN Solution

To set up a Xen High Availability solution, you need a SAN. In most cases, this would be a proprietary SAN solution, based on either Fibre Channel of iSCSI. In some cases, this is just not feasible, because a proprietary SAN is normally rather expensive. If that is the case for your environment, you might consider the Open Source SAN solution. This is an iSCSI SAN that is completely based on open source software. You can run it on any distribution you like. In this appendix you'll learn how to set it up on SLES.

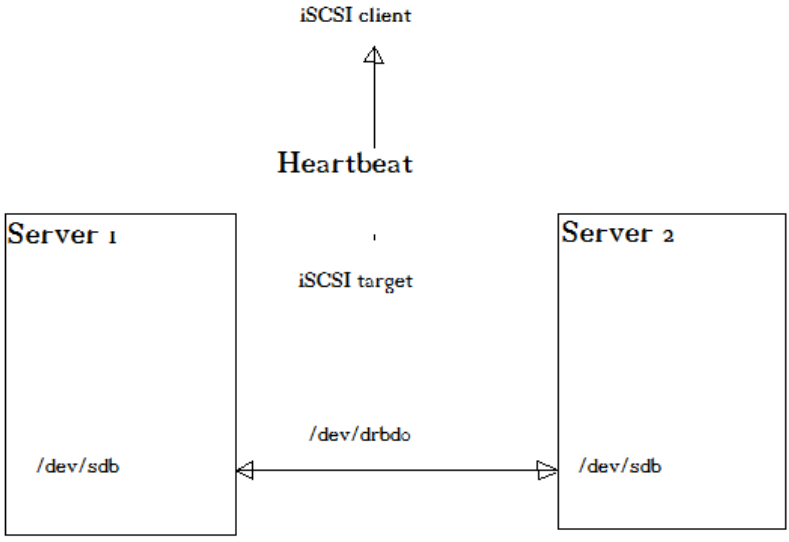
***Note**: During the final review phase of this book, I've had several requests to include some information about creating your own SAN solution in the book. Because of the very tight deadline, I haven't been able to write a completely new chapter. However, for the readers convenience, I've included this text, which is available as a download from my website also. The text is written for SUSE Linux Enterprise Server 10 sp2, but is applicable to SLES 11 as well. Based on the knowledge about Pacemaker that you've acquired in the first chapters of this book, you shouldn't have a hard time creating your own SLES 11 based open source SAN.*

9.1 Your own SAN based on DRBD, Heartbeat and iSCSI

Not everyone has money to buy a SAN appliance. Few companies however can afford not having a SAN and thus loosing critical company data when their servers storage crashes. You don't need a proprietary SAN appliance, but can build your own SAN using open source software as well. This article discusses how to do that. Before starting to build your own SAN solution, you should determine what exactly you need. Typically, a SAN consists of different components of which highly available storage is the core. If your SAN consists of just one box, you are playing with fire, if this box fails, your data will be unavailable and that will cost you money. So in real life, it takes two boxes to build a SAN and when building an open source SAN, this is not different. That is where the Distributed Replicated Block Device (DRBD) comes in. Using this solution, you can build your own RAID 1 across the network which means that if one of the servers that is providing storage goes down, the other server will take over this role immediately.

In a highly redundant SAN solution, you'll need one unique point of access. Some SAN solutions work with Fibre Channel. Because this involves an expensive fibre optics based infrastructure, such a solution is not the best you can imagine for an affordable open source SAN. Instead of using Fibre Channel, we'll use an iSCSI based solution on which the iSCSI target is used to provide access to the storage device.

One of the challenges of building this flexible SAN solution, is to keep it as versatile as possible. That means that a failing component has to switch over automatically. So if the SAN box that currently hosts DRBD as well as iSCSI access goes down, the other box in the SAN needs to take over automatically. To do this, we'll use Heartbeat to monitor and ensure availability of all cirtical resources in the SAN.



Schematical overview of the Open Source SAN

9.2 Required Software

To build such a SAN solution, you need the appropriate software. Typically, any Linux distribution will do, but in this appendix we'll use SUSE Linux Enterprise Server. This Linux distribution is available as a free download from http://www.novell.com/download.

One of the advantages of using SUSE, is that it is the distribution that is used by the developers of the Heartbeat software, that plays a critical role in this setup. You'll therefore always be ensured from optimal integration between the operating system and the cluster software when working with SUSE. You could use the open source version of SUSE Linux as well, but to build a stable and reliable setup, I would always recommend working with the enterprise version.

When making your choice for software you want to install, keep it to a minimum. After all, you are building a SAN solution here and typically on a SAN you wouldn't run any services. From the list of installable software patterns, select the following:

- Server Base System
- Documentation
- GNOME Desktop Environment for Server
- X Window System

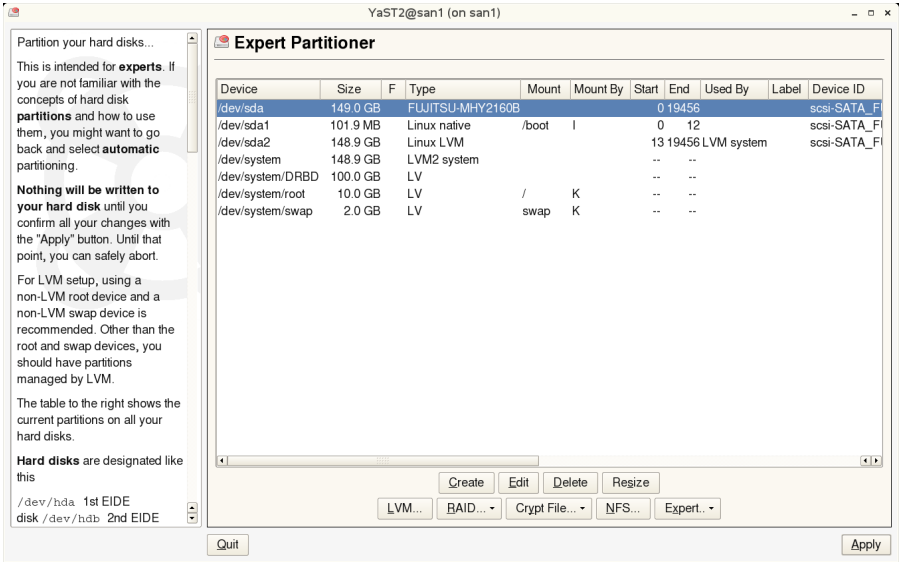
Also make sure that all the software from the High Availability Extension has been installed.

Apart from that, select the iSCSI target components as well, and you don't need anything else. In fact, you can even do it with less and choose not to select the GNOME environment and X Window System, but since SUSE heavily relies on YaST to configure the box, I'd recommend that at least for as long you are still working on the installation, you keep the graphical environment at hand - this is of course not necessary if you're fine working with the non-graphical version of YaST2 (run yast instead of yast2 if you've never seen it). Once finished the installation, you can still disable it. Of course, you're also free to start all graphical components from a desktop in case you would prefer that.

Apart from the software requirement, you should also think about the hard disk layout you want to use when creating a solution like this. No matter if you are setting up server hardware or two laptops to create a test environment, you should take into consideration that a storage device is needed as the SAN storage. In figure 1, I have used /dev/sdb as an example storage device, but this assumes that you have two (or even more) hard drives available. In case you don't have that, it may be a very good solution to create an LVM setup in which a large dedicated volume is made available for the SAN storage. The LVM setup also allows you to work with a snapshot solution later, more about that later in this article. On my test setup where two laptops with a 160 GB hard disk have been used, I've used the following disk layout:

- /dev/sda1: a 100 MB Ext2 formatted partition that is mounted on /boot.
- /dev/sda2: the rest of all available disk space, marked as type 8e for use in an LVM environment.
- /dev/system: the LVM volume group that uses the disk space available from /dev/sda2.
- /dev/system/root: a 10 GB Ext3 formatted logical volume for use as the root of the server file system.
- /dev/system/swap: a 2 GB logical volume used as swap space.
- /dev/system/DRBD: a 100 GB logical volume which is just allocated and not formatted.

To create such a disk layout, use the YaST integrated module while performing the installation.



To configure the disk layout of your SAN, YaST can be of great help.

The last part to take into consideration when setting up your server, is networking. I recommend putting your SAN on a network that is separated from normal user traffic. You don't want synchronization between the storage devices in the DRBD setup to be interrupted by a large file transfer initiated by and end-user, so if possible, create a dedicated storage network.

Normally, you would also want to configure a separated network for Heartbeat traffic, in order that a node doesn't get cast of the network when traffic is temporarily elevated. In this situation however, I would prefer not to do that. If Heartbeat packets don't come through over the normal network connection, it is likely that your DRBD device isn't communicating either anymore. You wouldn't want Heartbeat to ignore a failure in the communications link because a redundant link is still replying, while your SAN is in a disconnected stage would you?

9.3 Configuring DRBD

Assuming that your server is up and running according to the above mentioned installation guidelines, it is time to take the first real step in the SAN configuration and set up the DRBD device. Before starting this configuration, make sure that you have a storage device available on each of the servers involved in setting up your SAN. In this article, I'm working on the /dev/system/DRBD device, this name can be different in your particular configuration. It is also a good idea to use Ron Terry's add_drbd_resource script that you can download from:

http://pronetworkconsulting.com/linux/scripts/add_drbd_resource.html.
This script will create the configuration files automatically for you, this preventing you from making typing errors. The procedure below assumes that you are working with two servers that use the names san1 and san2. Replace these with the names of your servers if necessary.

1. Download the add_drbd_resource script from: http://pronetworkconsulting.com/linux/scripts/add_drbd_resource.html. Put it in user root's home directory and make it executable using the following command:

   ```
   chmod +x add_drbd_resource.sh.
   ```

2. Run the add_drbd_resource.sh script from san1 and provide the following values:
   Resource name: r0
   Number of the next DRBD device: 0
   Name of the block device to be mirrored: /dev/system/DRBD
   Name of the first server: san1
   IP address of the first server: (your server's IP address)
   Name of the second server: san2
   IP address of the second server: (your server's IP address)
   TCP Port number: 7788
   Syncer rate: This refers to the speed that is used when synchronizing changes between the DRBD devices. It can be as fast as the LAN connection between your servers. So use 100m on a 100 mbit LAN, or 1000m on a Gigabit LAN. Want to make sure that DRBD doesn't eat all available bandwidth? Set it somewhat lower.
   Synchronization protocol: use protocol C at all times because it gives you the optimal combination of speed and reliability.

After entering all required parameters, write the configuration to the /etc/drbd.conf file. This results in a file with a contents as in the file from the listing below:

Listing: Example of the drbd.conf file.

```
san1:/etc # cat drbd.conf
#
# please have a a look at the example configuration file in
# /usr/share/doc/packages/drbd/drbd.conf
#
# begin resource drbd0
resource drbd0 {
  protocol C;
        startup {degr-wfc-timeout 120;}
        disk {on-io-error detach;}
        net {}
        syncer {
                rate 100m;
                group 1;
                al-extents 257;
        }
```

```
        on san1 {
                device /dev/drbd0;
                disk /dev/system/DRBD;
                address 192.168.1.230:7788;
                meta-disk internal;
        }
        on san2 {
                device /dev/drbd0;
                disk /dev/system/DRBD;
                address 192.168.1.240:7788;
                meta-disk internal;
        }
}
# end resource drbd0
```

Now that on san1 the configuration is in place, you can load the DRBD kernel module using modprobe drbd. If this gives you a "module not found" error, make sure that you have installed the appropriate module for the kernel that you are using. (To find out which kernel that is, use uname -r). Now that the kernel module is loaded, it is time to run a first test. To do this, you use the drbdadm command with the -d (dry run) option and the adjust argument. This will make sure that the DRBD device is synchronized with the configuration file that you've just created. To do this, now run the following command:

```
drbdadm -d adjust drbd0
```

This command gives some results. If it looks like the output below, you are good to go and you can copy the configuration to the second server.

```
drbdsetup /dev/drbd0 disk /dev/system/DRBD internal -1 --on-io-error=detach
drbdsetup /dev/drbd0 syncer --rate=100m --group=1 --al-extents=257
drbdsetup /dev/drbd0 net 192.168.1.230:7788 192.168.1.240:7788 C
```

Now that the DRBD device is up and running, you can copy it to the other server:

```
scp /etc/drbd.conf san2:/etc/drbd.conf
```

Next, on the second node as well you need to load the DRBD module and test the DRBD device. So enter the following two commands from SAN2:

```
modprobe drbd
drbdadm -d adjust drbd0
```

If this all looks good, you can now go on and really start the DRBD device. First, you need to add the service to your current runlevels, using insserv drbd on both servers. Next, use rcdrbd start on both servers to start the service. This enables the DRBD device, but doesn't put it in a synchronized state. Now you need to enter the following command on one of the servers to tell this server that is should act as the primary (which is active) server:

```
drbdsetup /dev/drbd0 primary --do-what-I-say
```

The DRBD service is now up and running and the primary-slave relation should be established as well. Use rcdrbd status to test if it is really all working well.

Listing: Use rcdrbd status to test if the DRBD device is working well.

```
san1:/etc # rcdrbd status
drbd driver loaded OK; device status:
version: 0.7.22 (api:79/proto:74)
SVN Revision: 2572 build by lmb@dale, 2006-10-25 18:17:21
 0: cs:SyncSource st:Primary/Secondary ld:Consistent
    ns:2342668 nr:0 dw:0 dr:2350748 al:0 bm:12926 lo:0 pe:41 ua:2020 ap:0
        [>...................] sync'ed:  2.3% (99984/102272)M
        finish: 2:24:36 speed: 11,772 (11,368) K/sec
 1: cs:Unconfigured
```

As you can see from the listing above, the DRBD device is set up, but still synchronizing. This will take some time after initially creating the device. Give it some time, it will get synchronized completely after a while.

In this procedure you have read how to set up a DRBD device. That's a nice start if you want to create the open source SAN solution, but it's not good enough. As it is right now, if san1 goes down, you need to issue the drbdsetup primary command manually on the other server. Typically, you don't want to do that and that is what you are going to create a Heartbeat cluster solution for. In the next section you will read how to set up this cluster.

9.4 Setting up the Heartbeat Cluster

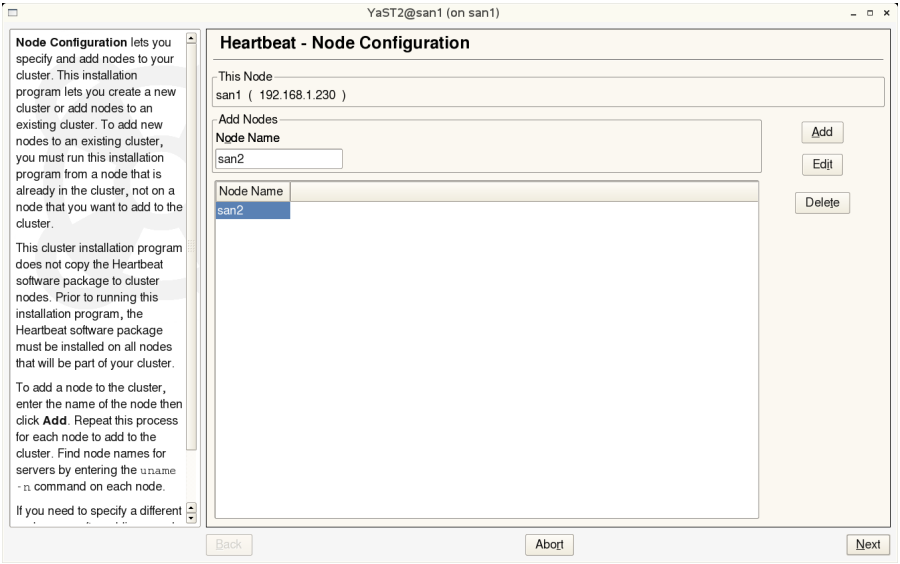
Let's be clear about this: setting up a Heartbeat Cluster is just something else. Doing it properly involves a lot of steps, such as setting up STONITH which ensures that a failing node is shut down automatically. I'm not covering all that, thus assuming that your Heartbeat network is already configured. In case it isn't, use the following minimal guidelines to create a simple two-node Heartbeat cluster in SUSE's YaST.

WARNING: *The procedure described below is a minimal procedure and for use in a test environment only. Use at your own risk if you want to set up a production environment using this procedure!*

1. Make sure that host name resolving is set up properly. If you are using DNS, probably everything is in place already. If not, make sure that the /etc/hosts file on all nodes include names and IP addresses of all other nodes.

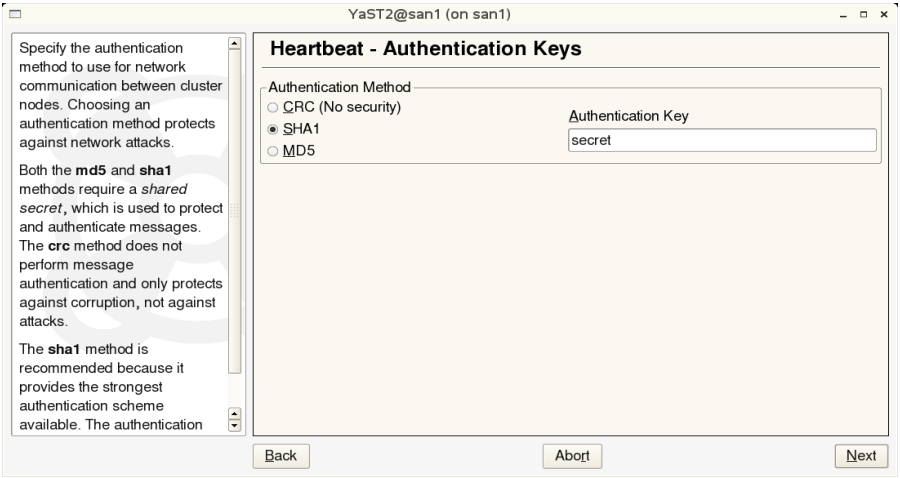
2. On san1, start the YaST administration utility, using yast2.

3. From YaST, select Miscellaneous > High Availability. In the window you see now, you already see the name of the node that you run YasT from. In the Add Nodes bar, enter the name of the node you want to add and click Add. This gives you a result as in the figure below. Now click Next to proceed.



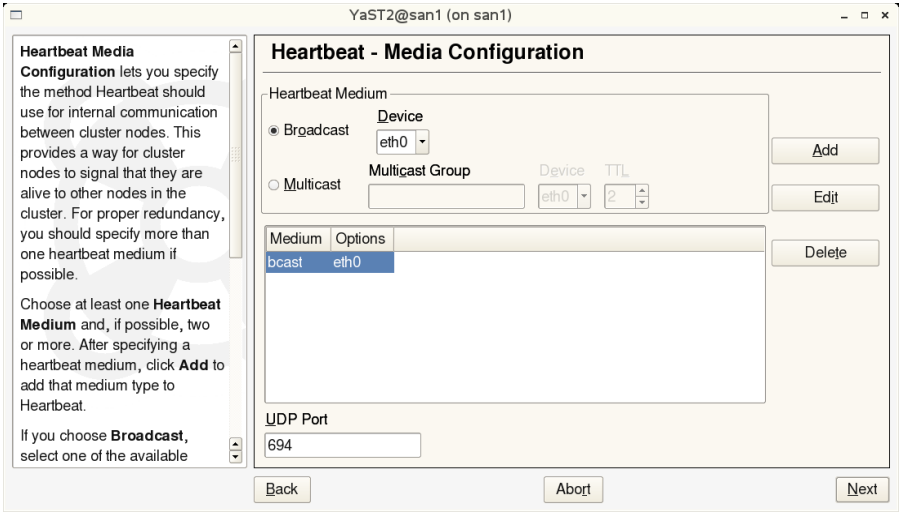
This is what the Node Configuration window should look like after adding all nodes.

4. Next, you need to select an authentication method. Strictly speaking, you don't need any authentication, but if you want to deploy more than one Heartbeat network in the same broadcast domain, it is a good idea to use either SHA1 or MD5 as the authentication method. By providing an authentication key with either of these protocols, you can prevent networks from mixing up by accident.



Provide an authentication key to prevent different Heartbeat clusters from mixing up by accident.

5. In the Media Configuration window, you specify what connection is to be used for the Heartbeat traffic. By default, the Heartbeat traffic will be send as broadcast over the default LAN connection. It is a good idea to use the connection that is used for the storage synchronization between the two parts of the DRBD device here, so make sure that device is selected and then click Next to continue. You can safely ignore any error messages that are displayed.



Make sure to select the storage network interface for the Heartbeat traffic.

6. In the final step of the procedure, you can specify if you want to start the Heartbeat service automatically when your server boots or not. Make sure that it is started automatically and click Finish to complete this part of the procedure.

7. To finalize the Heartbeat installation, use the /usr/lib[64]/heartbeat/ha_propagate command. This command will use scp to copy the Heartbeat configuration to the other nodes in the network. This command copies the configuration to the other node, but doesn't start Heartbeat on that node as well. To start Heartbeat and make sure it comes up after a reboot, use the following two commands on the other node:

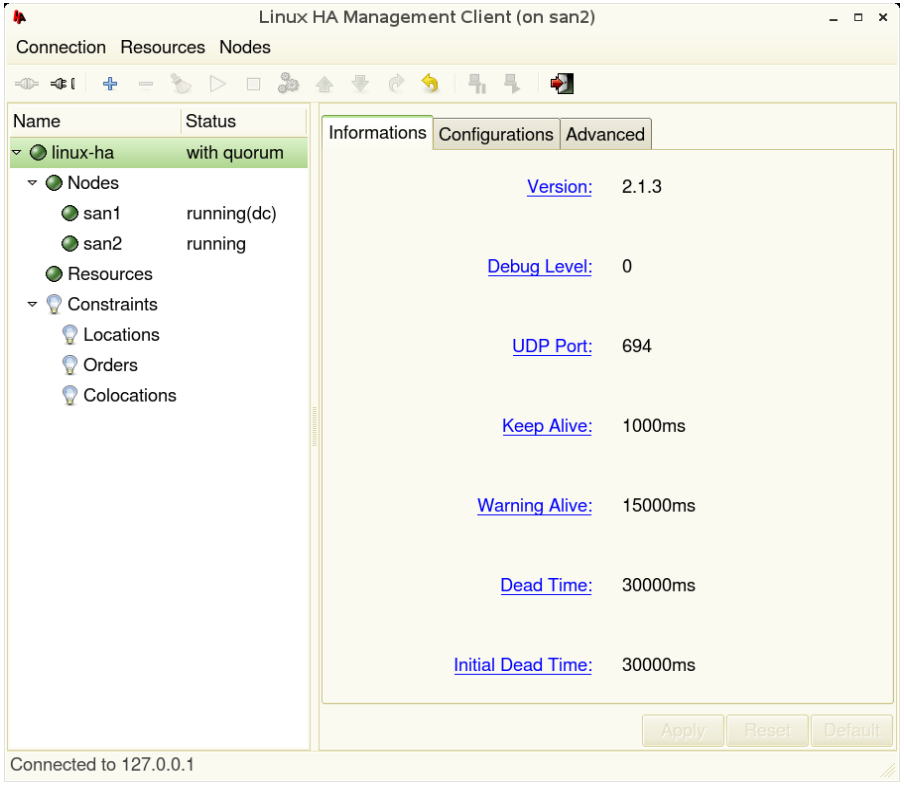
```
insserv heartbeat
rcheartbeat start
```

Your cluster is now up and running. There are two methods to do a quick check. The first is by using the crm_mon -I 1 command. This command shows you information about the amount of nodes that was found in the cluster and the associated resources.

Listing: The crm_mon command gives an easy method to see if nodes in your cluster are available.

```
san1:/etc # crm_mon -i 1
Refresh in 1s...

============
Last updated: Mon Jun 23 20:56:37 2008
Current DC: san1 (eecba864-30cf-4e47-9c1e-395381c3c460)
2 Nodes configured.
0 Resources configured.
============

Node: san1 (eecba864-30cf-4e47-9c1e-395381c3c460): online
Node: san2 (76d72cfb-efd3-4fde-975f-b8cafd5885bd): online
```

The other method to see if the cluster is up and running, is by using the hb_gui graphical user interface. To work from this interface, you need to provide the user hacluster with a password. So first, use passwd hacluster and next type hb_gui on either of the nodes in the cluster. After starting it, use Connection > Login to authentication. You'll now see the current status of the cluster in which at this moment there will only be two nodes and no resources.



From the hb_gui interface you can check the current status of the cluster.

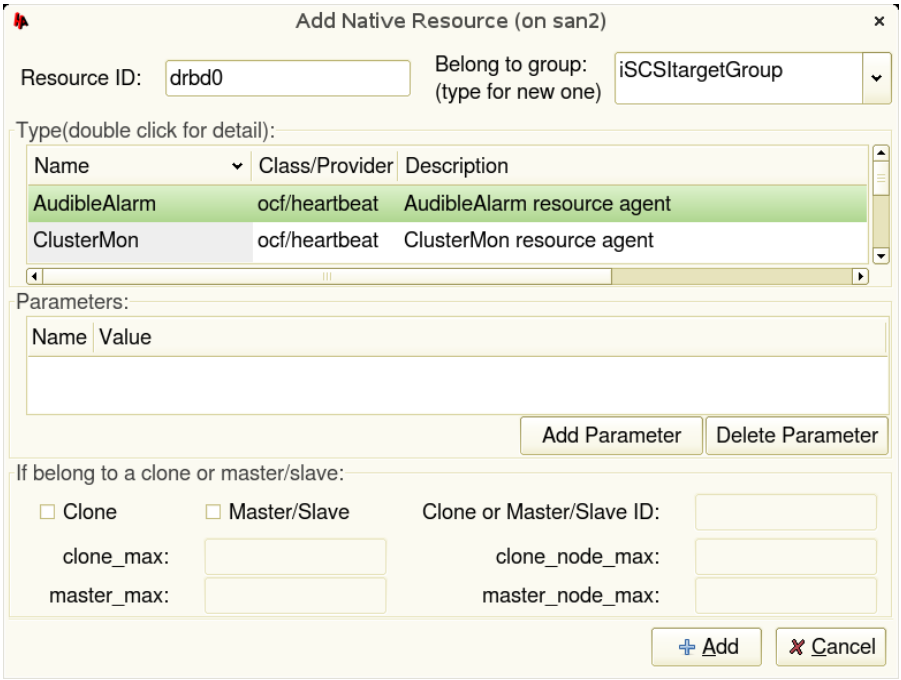
9.4.1 Configuring the DRBD resource in Heartbeat

You now have configured your server to start the DRBD service when booting. Normally in a Heartbeat cluster, that' s no good, but in this particular scenario, it is. All you need to do in Heartbeat, is configure the drbddisk resource. The current status is that there are two resources available for DRBD. The old drbddisk resource does work, the Heartbeat version 2 style OCF agent doesn't work, so in this document I'll describe how to use drbddisk. The only prerequisite for using this, is that DRBD must be started from the runlevels of all servers involved. Next, from the hb_gui interface, create the resource as described below.

1. From one of the nodes, start hb_gui and authenticate as user hacluster.

2. Select Resources > Add New Item and select the Item Type native. Click OK next.

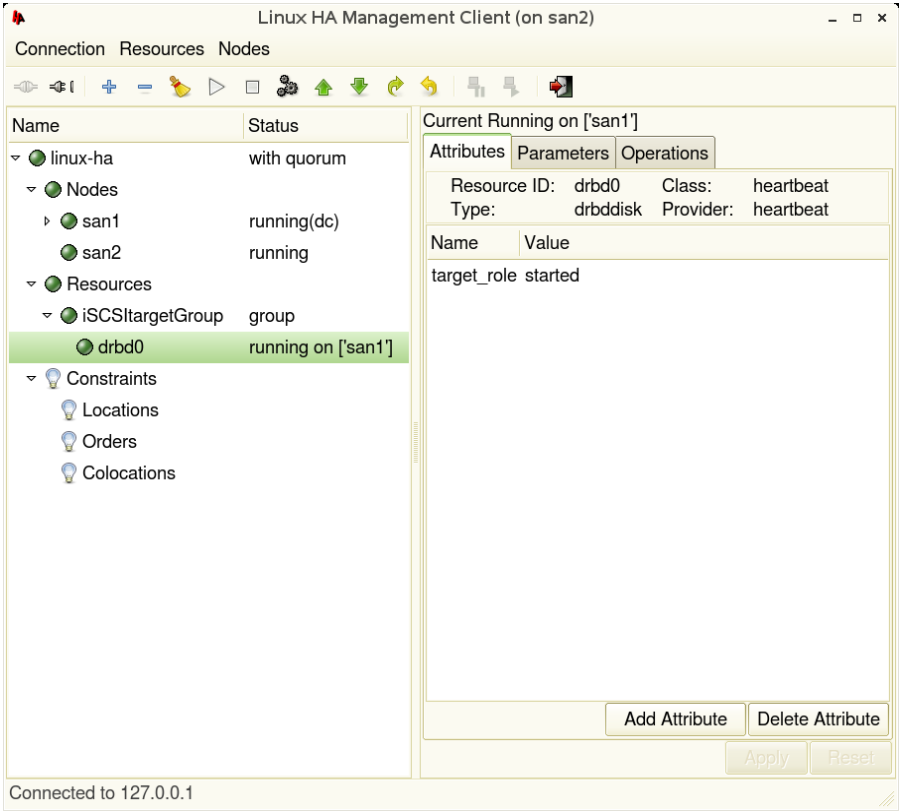
3. The ultimate goal is to create a resource group that manages DRBD as well as the iSCSI target service. To make this easier later on, you'll start working on the group right from the beginning. In the Add Native Resource window that you can see from the figure below, enter the resource ID drbd0 and at the option Belong to group type "iSCSItargetGroup". Don't click Add yet.



From the Add Native Resource window you specify what exactly the resource should do.

4. Now from the drop-down list of DRBD types, select the resource type with the name drbddisk. Next click Add Parameter and enter the value drbd0. This value should reflect the name that you have used when creating the DRBD device in the /etc/drbd.conf script. Next, click OK and Add to add the resource to your cluster configuration.

5. In the hb_gui interface, you'll now see that the drbd0 resource has been added, but has the status of not running. Right-click the resource and click Start to start it. You should now see that the resource is running on one of the two nodes and that is fine.



The hb_gui now indicates that the DRBD resource has been started on one of the nodes in the cluster.

The DRBD resource is now running on one of the cluster nodes. In the figure below, you can see it is currently being served by san1. Time for a stress test: when you pull the plug from san1, the DRBD resource should fail over automatically. Before doing this, use watch cat /proc/drbd to monitor the current status of the DRBD resource on node san2. You should see that it currently has the status

secondary, but once the cluster has failed the resource over, the status should change to primary.

Listing: From /proc/drbd you can monitor the current status of the DRBD resource.

```
san1:/etc # watch cat /proc/drbd
Every 2.0s: cat /proc/drbd                                  Mon Jun 23 21:43:48
2008

version: 0.7.22 (api:79/proto:74)
SVN Revision: 2572 build by lmb@dale, 2006-10-25 18:17:21
 0: cs:Connected st:Primary/Secondary ld:Consistent
    ns:104726528 nr:0 dw:0 dr:104726528 al:0 bm:19176 lo:0 pe:0 ua:0 ap:0
 1: cs:Unconfigured
```

Everything in place for the stress test? Time to start the test: from the san1 node, you are going to kill the cluster now by using the pkill heartbeat command. This will terminate the Heartbeat service on node 1 and it will make the other node primary in the DRBD seteup.

Everything working so far? Very good. In that case DRBD is up and running. In case you are seeing instable behavior, it is a good idea to do a restart of both nodes. This will bring up DRBD right from the start as a resource that is managed by Heartbeat and that increases the chances of success. If this doesn't work, then don't proceed to the next step. If it does work, it's time to configure iSCSI.

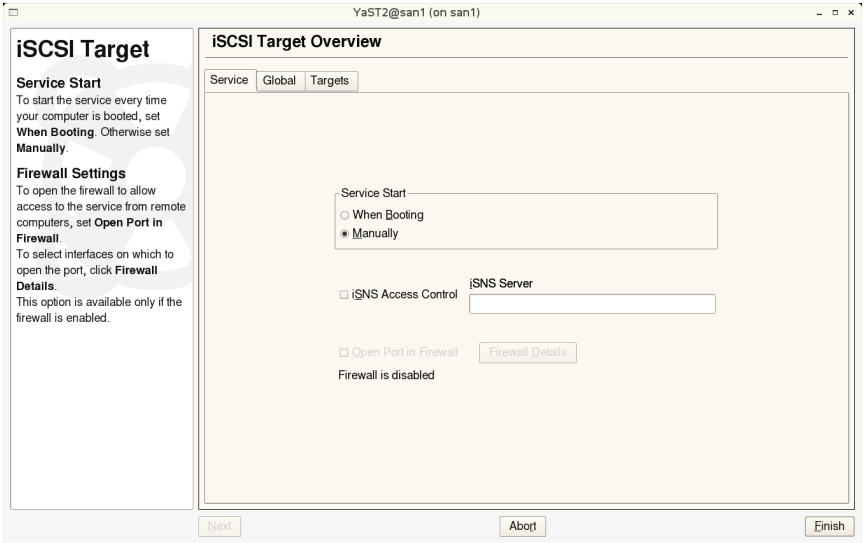
Note: *After restarting the servers in your cluster, it will take some time before crm_mon -I 1 shows both nodes as up and available again. That normally is no problem, just wait for the nodes to come up before proceeding.*

9.5 Configuring iSCSI

There are in general two different types of SAN; iSCSI based SAN's and Fibre Channel based SAN's. A Fibre Channel SAN requires expensive hardware and therefore is not an option for the open source SAN. In this configuration, we'll be using iSCSI. In a typical iSCSI configuration, the iSCSI target is the service that is provided by the SAN. This iSCSI target provides SCSI access to a shared device. In this case, the shared device will be the drbd0 device that you have created earlier and which is made highly available in the preceding step of this procedure. Configuring iSCSI is not a very difficult two step procedure. First, you'll use YaST to set up the iSCSI service and after that, you'll make the iSCSI target service highly available.

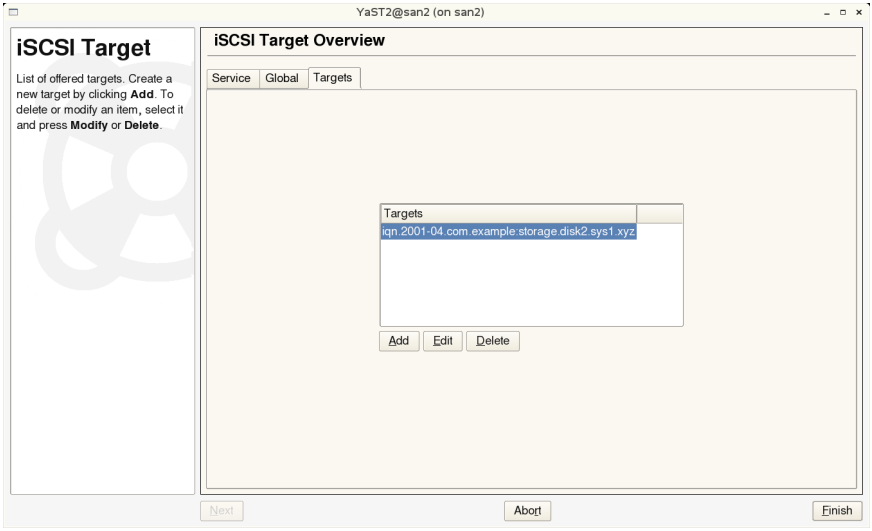
1. From on of the servers in your cluster, start YaST by entering the yast2 command (if you prefer clicking, you could also select it from the start menu of the server of course, but first, servers normally are in cold data centers and system administrators prefer nice and warm, and second, a typical Linux server doesn't run a GUI, does it? So start you SSH session, and from there, enter the yast2 command, okay?)

2. Select Miscellaneous > iSCSI Target. This may give a prompt that some software must be installed first. If that's needed, do it. Once done, you'll see an interface as in the following figure.



The YaST iSCSI Target Overview window offers an easy interface to configure iSCSI.

3. On the Service tab, make sure that the option Manually is selected. This is because the cluster is going to manage the iSCSI target startup, therefore you must make sure that it will not be started automatically. Next, select the targets tab . On this tab, you'll see an example iSCSI target. It is not functional, so select it and click Delete to remove it.



From the Targets tab, remove the example iSCSI target.

4. Now click Add to add a new iSCSI target. The target name is selected automatically, which is fine. For the identifier a unique ID that is generated automatically is used. Provide a more readable ID here; you can pick any name you like. Next, click the Add button. The LUN number you have to enter here, must be a unique ID for the target. Next, make sure that the option Type=fileio is selected and browse to the DRBD device /dev/drbd0 to configure that as the iSCSI target storage device. No other information is required here, so click OK to close this window en then click Next to proceed.

Configure the iSCSI target to share the DRBD device.

5. In the authentication settings window, no settings need to be changed. Click Next to proceed and when back in the Overview window click Finish to write the configuration to your server.

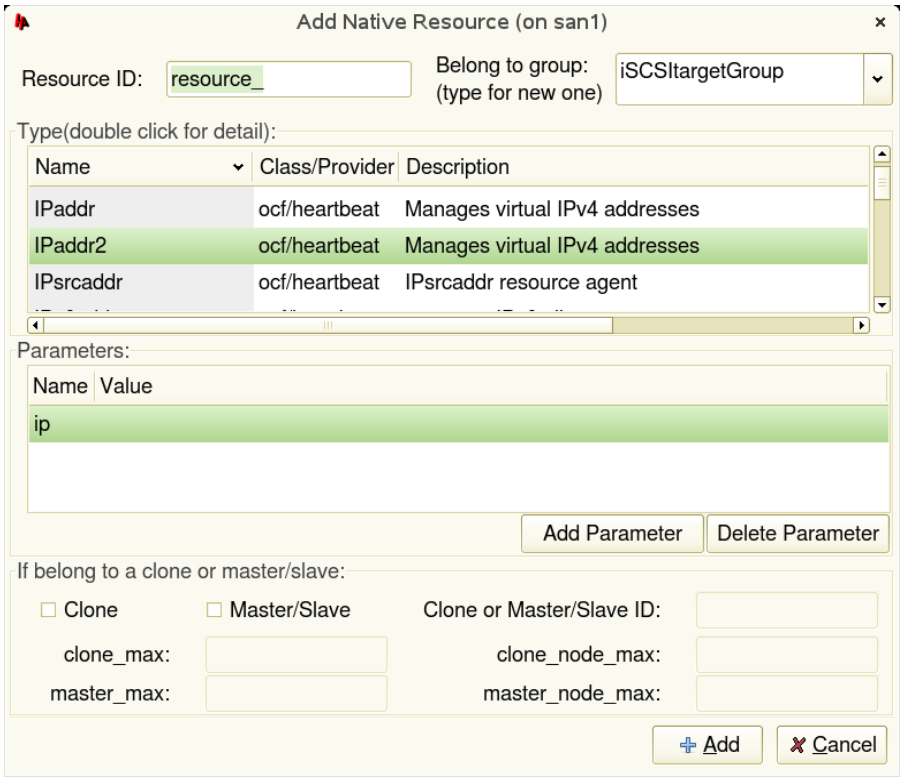
6. After creating the iSCSI target configuration on the first server, you need to make sure that the second server has the same configuration. You can ensure this by copying the /etc/ietd.conf file to the other server. So assuming that san2 is the other server, use the following to copy the configuration file from a san1 console:

```
scp /etc/ietd.conf root@san2:/etc/ietd.conf
```

The status at this point is that the iSCSI target is useable, but will not be started automatically. You have to add it to the cluster now. To do that, two different components are needed: a unique IP address on which the iSCSI target can be used and a resource that will start the iSCSI target service for you. The following procedure describes how to add these:

1. Start the hb_gui utility from one of the servers.

2. Select the iSCSItargetGroup you've created earlier, right click on it and from the drop-down list, select the Item Type native. Next, enter the Resource ID iSCSI_IP_Address and from the Type list, select ipaddr2. In the parameters section of the interface, you'll see that the ip parameter has been added automatically. Click in the Value column to add an IP address here. Next enter a unique IP address.



You need to enter a unique IP address for the iSCSI target.

3. Now click the Add Parameter button and select the parameter cidr_netmask. In here, enter the netmask your IP address has to use. You need to enter this in CIDR format, not in dotted decimal format. So 24 and not 255.255.255.0. Next click OK to add it. In case your server has multiple network boards, you also want to make sure that the IP address binds to the right network board. To accomplish this, click Add Parameter once more and from the drop down list at the name option, select nic. Next enter the name of the interface to which you want to assign the IP address and click Add to add the resource.

4. Apart from the unique IP address to be used by the target, you need to create a resource for the iSCSI service as well. Right-click on the iSCSItargetGroup and

from the menu, select Add New Item. When asked what kind of resource you want to add, make sure to select the Native resource type.

5. As the name of the resource, enter iSCSI_target and from the list of available resource types, select iscsitarget (lsb). Next click Add to add the iSCSI target resource.

6. Everything is in place now to start the three associated resources. Right click the iSCSItargetGroup and from the menu, select Start. This will start the resources and since they are in the same resource group, all will be started on the same node. Congratulations, you now have your open source SAN solution ready to go!

9.6 Connecting to the SAN

Now that the SAN is in place, all you need to do, is connect to it. To do that, from the server that you want to connect you need to initiate an iSCSI session. You can do that by using a software iSCSI initiator as they are available for almost all server operating systems. Alternatively, you can use an iSCSI HBA as well. For instance, qlogic HBA's are quite popular. These contain an integrated iSCSI server that will connect to the SAN. Using a hardware HBA is a good idea, because it will absolutely give you the best performance. This is because the iSCSI process has its own dedicated chip, so it doesn't have to compete with other system resources for CPU cycles and memory. Configuring the iSCSI connection to the SAN falls beyond the scope of this article, just read one of those articles that says you need to connect to the SAN. You now know how to do that.

You should also know what exactly you can expect from your Open Source SAN Solution. Of course, you can expect availability. But what if the active SAN server goes down? In that case the iSCSI initiator will lose its session. That is something that just comes with iSCSI, no way to avoid that. So if the active SAN node fails, the other node will take over fast enough. The servers that are connected to the SAN however will need to re-establish their connection to get into contact again. So what you reach, is high availability, and that can never be the same as uninterrupted availability.

Alphabetical Index

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

R

S

U

V

W

X

Y

Z