

Configuring Xen High Availability

Heartbeat at SUSE Linux Enterprise Server

The most important improvements in SUSE Linux Enterprise Server 10 support pack 1, are related to Xen virtual machine management and Heartbeat high availability clustering. Combined together they are capable to compete with expensive VMware functionality for high availability; Heartbeat clustering can be used to make Xen virtual machines highly available.

The advantage of the combination between Xen and Heartbeat is that with this solution, you won't lose virtual machines anymore. Everything that is needed to build such a solution, is included in SUSE Linux Enterprise Server. In this article you can read how to configure it.

Before we start, let's make a small list of required components:

- * A storage volume where all nodes involved in the cluster can write to at the same time.
- * A file system that allows for simultaneous writes
- * The Heartbeat software for high availability clustering
- * One or more Xen virtual machines

Configuring the SAN

The first part of the installation of an environment for Xen virtual machines high availability (we'll refer to that as Xen HA for the rest of this article), involves creating a SAN. The reason for that is that the Xen disk images and configuration files have to be stored on a location where they can be reached by both nodes simultaneously. If you already have a SAN solution in place, you can use that. Create two LUN's and continue reading the next section in that case. If you don't yet have a SAN, SUSE Linux Enterprise Server (SLES) 10.1 includes everything you need to create one based on iSCSI. In this article, we'll describe a solution where one server running SLES 10.1 is the storage server and two servers running SLES 10.1 access the LUN's that are offered by the storage server.

1. On the storage server, you need a device to share. This can be a complete hard disk, a partition or logical volume, or a file that is created as a disk image file. Since in this scenario you'll need at least 4 Gigabytes to store the Xen virtual disk image, for performance reasons, I recommend using a real device such as a partition or volume. Make sure this device is available before you proceed. In case you don't have the opportunity to use a real device, you can use **dd if=/dev/zero of=/var/xenimages bs=1M count=4096** to create a 4 GB file to be used for storing the Xen disk image. Next, make sure you have a small device available for storing the Xen virtual machines configuration files as well. For example, use **dd if=/dev/zero of=/var/xenconfig bs=1M count=1024** to create this as a 1 GB disk image file.
2. Next, at the storage server you have to configure an iSCSI target that shares the disk devices. Start *YaST*, enter the password for the user root and from *Miscellaneous*, start the *iSCSI Target module*. At the *Service* tab, make sure that *When Booting* is selected. Next, on the targets tab delete the example target that exists already. Proceed by clicking *Add* to add a new target. This brings up a window like the one in Figure 1.

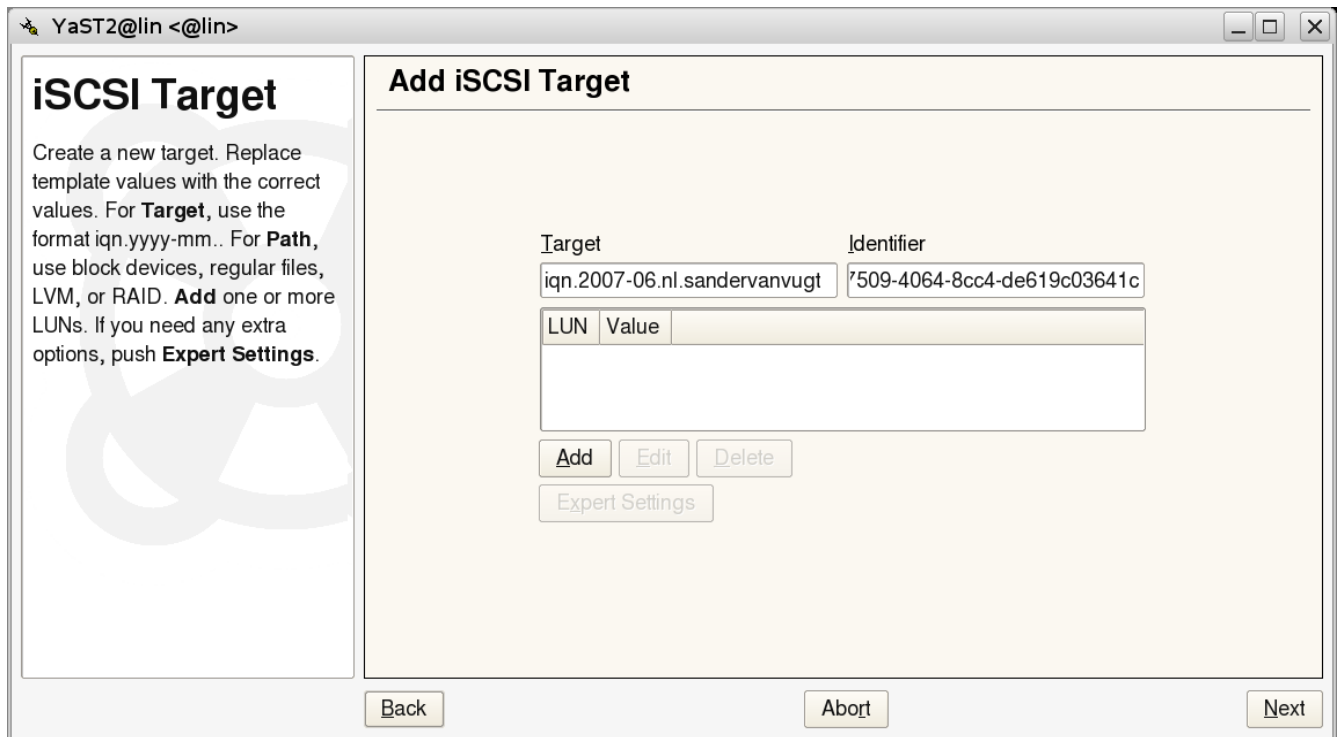


Figure 1: From here you specify what LUN's to offer with the iSCSI target server.

3. Now, click *Add* to define the LUN's themselves. Every LUN is offered as a device on itself at the nodes in the cluster and for every LUN that you create, you need a separate device to share. Therefore, select the LUN number for your first LUN and in the Path field, specify the name of the device that you want to share, for example: `/var/xenimages`. Next click *OK* to add the device and close the iSCSI Target configuration program.
4. Now that your storage server is offering the shared storage, go to the console of the nodes that are going to use it and from there, start the *iSCSI Initiator* module from *YaST*. At the *Service* tab, make sure that *When Booting* is selected. Next, select the *Discovered Targets* tab and from there, use the *Discovery* button. Enter the IP address of the iSCSI target server, leave the authentication options blank and click *Next*. This brings up a window like in Figure 2 where the name of the iSCSI target service is mentioned. Click the link and then select *Log In*, to make sure that you are connected.

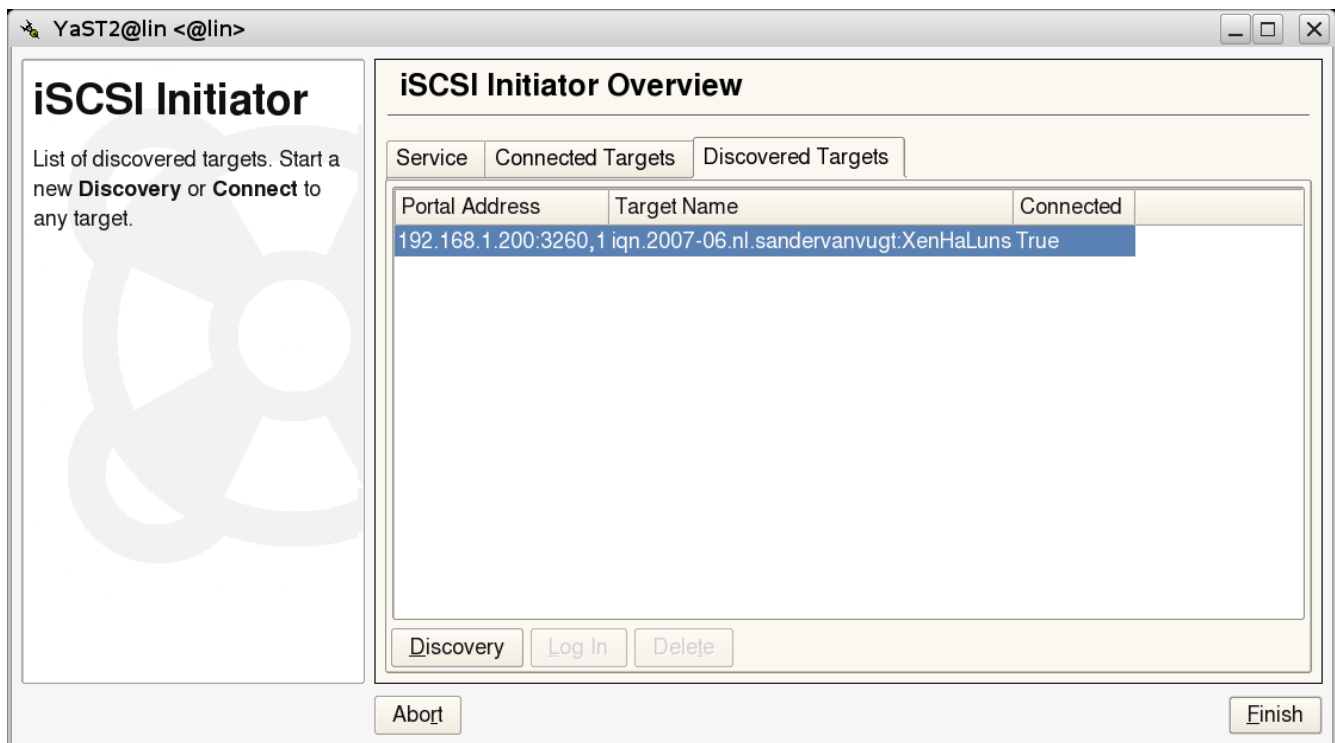


Figure 2: Make sure that you are connected to the iSCSI target before proceeding.

5. Now click the *Connected Targets* tab and from there, click *Toggle Start-Up*. This ensures that the connection is established automatically the next time that your server reboots. Click *Finish* to complete the wizard and repeat step 4 and 5 on all other nodes in the network that need access to the shared storage.
6. To verify that it is really all working fine, you can use the `lsscsi` command at all nodes in the cluster. This command should list two new disk devices that are available. Also, you can use the `iscsiadm -m session` command to display the session that exists from the initiator to the target.

Configuring the cluster safe file system

Now that the SAN is in place, it's time for the second step: configuring a cluster safe file system. This is a file system that can be written by multiple nodes simultaneously. Currently, there are two major players that can be used as cluster safe file system; Red Hat's Global File System (GFS) and Oracle's Oracle Cluster File System version 2 (OCFS2). Since OCFS2 is included in SLES 10.1, we'll use it in this article. Before we start, there is one thing that you should know about OCFS2. Typically, it has its own high availability solution. In this case however, we are not going to use that because we want the OCFS2 volumes to be managed by Heartbeat. This requires some additional steps, which are explained below.

1. At the console of one of the nodes in your cluster, run the `ocfs2console` command to start the graphical interface to create and manage your OCFS2 environment. From this interface, select *Cluster, Configure Nodes* and add all the nodes that are in your cluster to the list. Make sure that the node name you are using is the same as the result of the `uname -n` command.

2. When finished adding nodes to the cluster, open the *Cluster* menu and use *Propagate Configuration* to copy the configuration to the other nodes in the cluster. To do this, SSH is used, so be sure that it is configured properly.

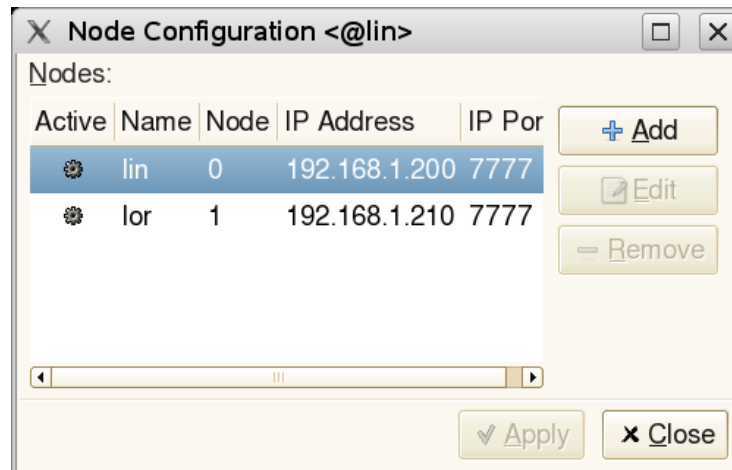


Figure 3: The *ocfs2console* offers an easy interface to configure the cluster that is needed to use an OCFS2 file system.

3. Now on both nodes, you have to tell the node that it should use Heartbeat instead of the OCFS2 cluster solution. To do that, run the **rc02cb configure** command on both nodes. Accept all default answers, but when it asks if you want to use user space heartbeat, answer Y instead of N. Next, only on the node where you have used the *ocfs2console*, use the **rc02cb force-reload** command to reload OCFS2 properly.

4. Before we finish this part of the OCFS2 configuration, you have to tune one setting in the configuration file `/etc/sysconfig/o2cb` and that is the parameter **O2CB_IDLE_TIMEOUT**. By default this parameter has a timeout of 10 seconds in which it has to establish a communication with the other node. I recommend you to increase that, otherwise you'll have the risk that the OCFS2 cluster fails. If it still fails, increase the other timeout values in this file as well. It will make the OCFS2 file system a bit slower, but helps you as well in being more succesful setting this up. And after all, when it works, you can always tune these parameters some more later.

Configuring Heartbeat

Now that OCFS2 is configured the right way, you have to set up Heartbeat. This is the most important part of this setup, since it is the part that ensures that your service is going to be available at the location where you need it. There are several parts involved. First, you have to create the cluster itself. Once the cluster is up and running, you have to configure the cluster resources for the OCFS2 file systems that you want to use. This ensures that the OCFS2 file system will be running at all times on all nodes where you need it. After that, you need a cluster resource for every single Xen virtual machine that you want to manage. Finally, you need to tell the cluster that it always must load the OCFS2 file systems before starting the Xen virtual machine. Let's start with the configuration of the Heartbeat Cluster itself:

1. There is some work to be done before you start the cluster configuration: it is very useful to configure the following:

- * SSH key-based authentication

- * NTP time synchronization
- * The atd service should run on all nodes in the cluster.

As the last part of the preparation, make sure that host name resolving is set up properly, for example by including the names of all nodes in the cluster in the `/etc/hosts` file on all nodes.

2. Once the preliminary steps mentioned in (1) have been completed, from **YaST**, select *High Availability*. In the *Node Configuration* window, make sure that all nodes that you want to be involved in the cluster are listed.

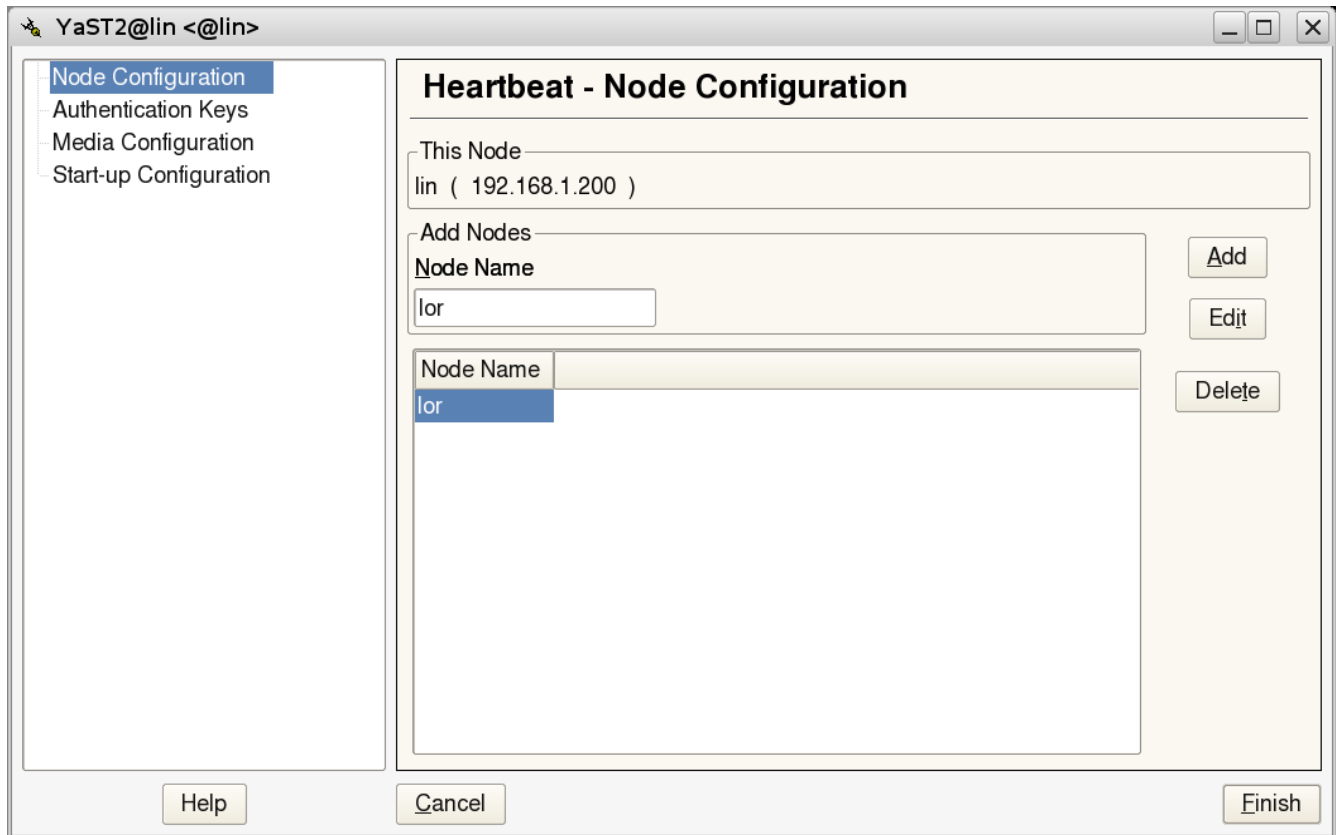


figure 4: All nodes involved in the cluster must be added to this list.

3. Next, for a simple test configuration you can accept the default values in the *Authentication Keys* and *Media Configuration* windows. In the *Start-up Configuration* window, make sure that the Heartbeat service is selected to start automatically when booting the server and then click *Finish* to write all the configuration.

4. Now you need to copy the configuration you have just created to all other nodes in the cluster. You can do this by running the `/usr/lib/heartbeat/ha_propagate` command at the console of the server where you've just created the cluster. On all other nodes, use the `rheartbeat start` command to start the Heartbeat service now.

5. The cluster should be up and running now. To verify this, use the `crm_mon -i 1` command on the console of one of the nodes involved in the cluster. After a maximal waiting time of one minute, it should indicate that the nodes are up and running. You now see a result as in the listing below:

=====

Last updated: Mon Jun 25 14:22:49 2007
Current DC: lin (00360938-b716-4308-82fd-e30878afa8ed)
2 Nodes configured.
1 Resources configured.
=====

Node: lin (00360938-b716-4308-82fd-e30878afa8ed): online
Node: lor (5d217a35-d976-48db-8daa-99473cfc6c11): online

Creating a STONITH resource

Now that the cluster is up and running, we can start with the real work and add some cluster resources. The first resource that we are going to create, is the STONITH resource. This resource (the acronym stands for “Shoot The Other Node In The Head”) makes sure that when the cluster thinks a resource is dead, it will make sure that it is dead. This is a very important part of the configuration, because it ensures that the shared file system cannot be corrupted. Normally, STONITH is implemented with a special device like a power switch, which just pulls the powerplug for a node that has to be terminated. If such a switch is not available, you can use the SSH-stonith device instead. This device, which was created for test purposes, sends the shutdown command to another host. In order to execute this command, the `atd` process must be running on all nodes.

For management of a Heartbeat cluster, two interfaces are available. First of them is the **hb_gui** graphical interface. This is the most convenient interface, because it offers an easy interface from which you can choose all elements that you need. To start with however, we'll use the more difficult way in which the **cibadmin** command is used to add code from XML-files to the **cib.xml**, which is the heart of your cluster.

1. Make sure the **atd** process is running. You can do this by issuing the **insserv atd** command which ensures that it will be started automatically the next time your server restarts. Next, make sure it is started by running the **rcatd start** command.
2. In general, there are two ways to add information to the cluster. You can get quite far by using the **hb_gui** graphical interface. Some options however are not available from the graphical interface and can be configured only by inserting an XML file that contains the configuration that you want to use into the cluster. In this article we'll show both ways. First, you'll learn how to create and add an XML-file and later you'll see how the GUI can be used. Let's first create an XML file that contains the generic properties that are used by the cluster. Create the file with the name **cibbootstrap.xml** and make sure it has the following contents:

```
<cluster_property_set id="cibbootstrap">
<attributes>
  <nvpair id="bootstrap-01" name="transition-idle-timeout" value="60"/>
  <nvpair id="bootstrap-04" name="stonith-enabled" value="true"/>
  <nvpair id="bootstrap-05" name="stonith-action" value="reboot"/>
  <nvpair id="bootstrap-06" name="symmetric-cluster" value="true"/>
  <nvpair id="bootstrap-07" name="no-quorum-policy" value="stop"/>
  <nvpair id="bootstrap-08" name="stop-orphan-resources" value="true"/>
  <nvpair id="bootstrap-09" name="stop-orphan-actions" value="true"/>
  <nvpair id="bootstrap-10" name="is-managed-default" value="true"/>
  <nvpair id="bootstrap-11" name="default-resource-stickiness" value="INFINITY"/>
</attributes>
```

```
</cluster_property_set>
```

3. Next, you need another XML file that defines the Stonith resources. Create a file and give it the name **stonithcloneset.xml** and make sure it has the following contents:

```
<clone id="stonith_cloneset" globally_unique="false">
  <instance_attributes id="stonith_cloneset">
    <attributes>
      <nvpair id="stonith_cloneset-01" name="clone_node_max" value="1"/>
    </attributes>
  </instance_attributes>
  <primitive id="stonith_clone" class="stonith" type="external/ssh" provider="heartbeat">
    <operations>
      <op name="monitor" interval="5s" timeout="20s" prereq="nothing" id="stonith_clone-op-01"/>
      <op name="start" timeout="20s" prereq="nothing" id="stonith_clone-op-02"/>
    </operations>
    <instance_attributes id="stonith_clone">
      <attributes>
        <nvpair id="stonith_clone-01" name="hostlist" value="LIN,LOR"/>
      </attributes>
    </instance_attributes>
  </primitive>
</clone>
```

4. Next, you need to add the contents of these two files to the cluster. This causes the configuration to be written to the **cib.xml**, which is the heart of the cluster that contains all the configuration. Do this by using the following two commands:

```
cibadmin -C -o crm_config -x bootstrap.xml
cibadmin -C -o resources -x stonithcloneset.xml
```

5. Now use the **crm_mon -i 1** command once more. You should see that the Stonith resources are added to the cluster, the listing below shows what you should see now.

```
=====
Last updated: Wed Jul 4 17:12:37 2007
Current DC: lor (5d217a35-d976-48db-8daa-99473cfc6c11)
2 Nodes configured.
1 Resources configured.
=====
```

```
Node: lin (00360938-b716-4308-82fd-e30878afa8ed): online
Node: lor (5d217a35-d976-48db-8daa-99473cfc6c11): online
```

```
Clone Set: stonith_cloneset
  stonith_clone:0 (stonith:external/ssh): Started lin
  stonith_clone:1 (stonith:external/ssh): Started lor
```

Creating the OCFS2 file system resources

Next, you need to create the OCFS2 file system resources. This time, we'll use the `hb_gui` graphical interface to do the job. You'll see that to prepare properly, we first need to grant a password to the user `hacluster`, which is the useraccount that you would typically use to do your work from the GUI interface. Also, we need to make sure that the resource that is created, is a resource that can be active at more than one node at the same time. We do this by using the so called clone resource. The following procedure describes how to create the OCFS2 file system resources.

1. From the command line, use the command `passwd hacluster` to give the user `hacluster` a password. Do this at all nodes where you want to run the `hb_gui` interface.
2. Start the `hb_gui` by entering `hb_gui &` on a console. Select *Connection, Login* to open the login window and authenticate as the `hacluster` user. This shows the window that you can see in figure 5.

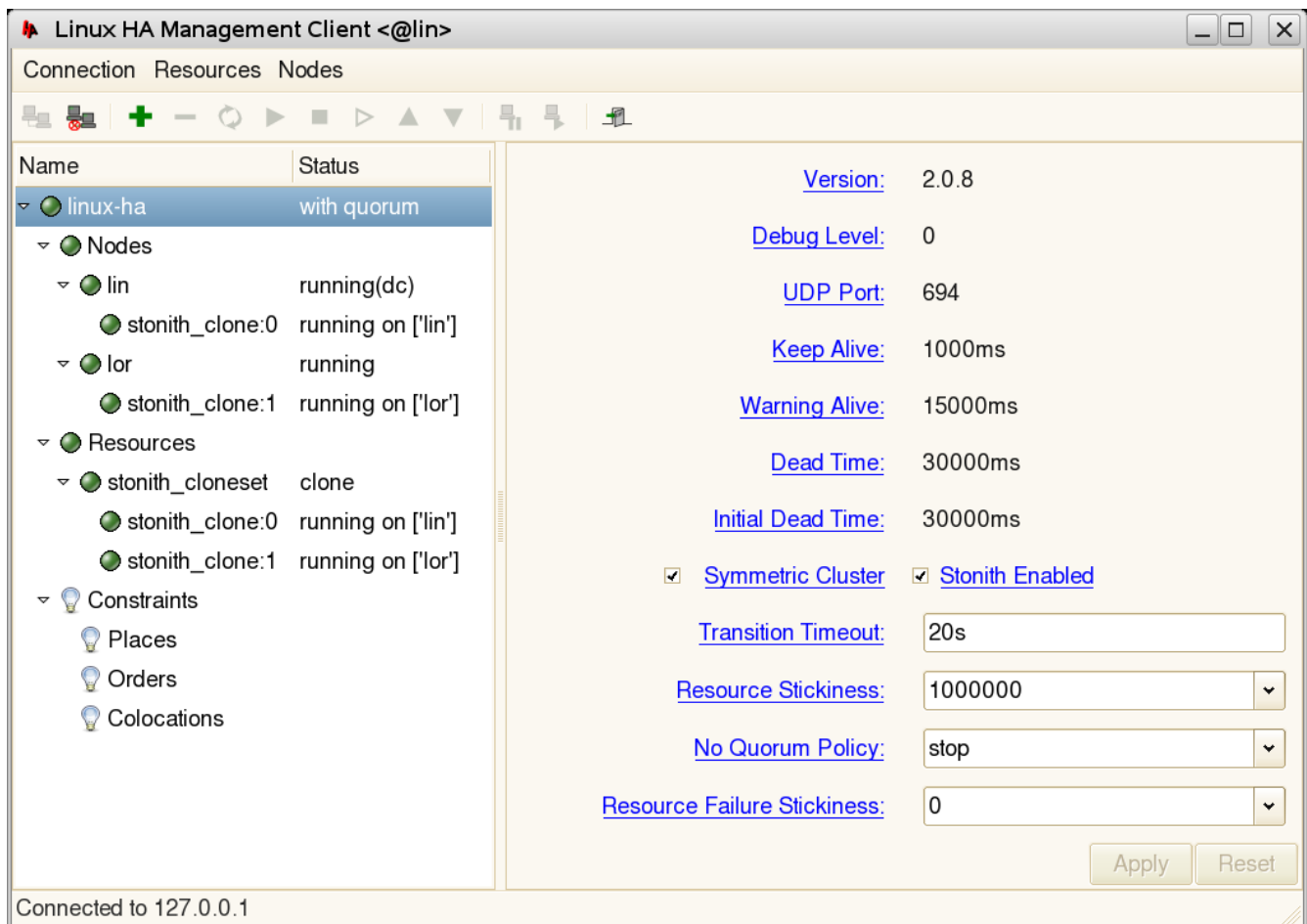


Figure 5: The `hb_gui` interface allows for easy management of Heartbeat

3. We are going to create cluster resources for shared file systems. The only way to be sure about the file system you are using, is by using the `/dev/disk/by-id` name of the device; a SAN device that can be presented as the second device at node1, can be the third device at node2 and therefore classic device names like `/dev/sdb` are not the ideal way. Therefore, open a console and from that console, activate the `/dev/disk-by-id` directory and use the command `ls -l` to show the UUID's of the disk

devices that are known to your server. Next, use **fdisk -l** to check the properties of these disk devices, so that you know what device to use for what purpose. You can see the result of this command in the listing below:

```
lin:/dev/disk/by-id # ls -l
total 0
lrwxrwxrwx 1 root root 10 Jun 25 2007 ata--part1 -> ../../sda1
lrwxrwxrwx 1 root root 10 Jun 25 2007 ata--part2 -> ../../sda2
lrwxrwxrwx 1 root root 9 Jun 25 2007 ata-WDC_WD1200BEVS-75RST0_WD-W XEX06088661
-> ../../sda
lrwxrwxrwx 1 root root 9 Jun 25 14:56 scsi-14945540000000000000000000000000
01000000e0800000d000000 -> ../../sdc
lrwxrwxrwx 1 root root 9 Jun 25 14:56 scsi-14945540000000000000000000000000
0100000025080000d000000 -> ../../sdd
lrwxrwxrwx 1 root root 9 Jun 25 14:56 scsi-14945540000000000000000000000000
01000000f8070000d000000 -> ../../sdb
lrwxrwxrwx 1 root root 9 Jun 25 2007 scsi-SATA_WDC_WD1200BEVS-_WD-W XEX06088661
-> ../../sda
lrwxrwxrwx 1 root root 10 Jun 25 2007 scsi-SATA_WDC_WD1200BEVS-_WD-W
XEX06088661-part1 -> ../../sda1
lrwxrwxrwx 1 root root 10 Jun 25 2007 scsi-SATA_WDC_WD1200BEVS-_WD-W
XEX06088661-part2 -> ../../sda2
```

In this example, we need **sdd** for storage of the configuration files for the Xen virtual machines, and **sdc** is used to store the image files.

4. Before making the cluster resources, you need to format the file systems with OCFS2. To do this, the complete disk ID must be used. In our case, the following two commands need to be issued:

```
mkfs.ocfs2 /dev/disk/by-id/scsi-14945540000000000000000000000000100000025080000d000000
mkfs.ocfs2 /dev/disk/by-id/scsi-149455400000000000000000000000001000000e0800000d000000
```

5. Now from the hb_gui interface, we need to create the clonesets for both of these disks. First, select *Resources*, *Add New Item* and select the resource type *Native*.

6. You now see the window that you can see in Figure 6. From here, you are going to create the cloneset. First, let's make the *Xenconfig_clone cloneset*, using the following properties:

```
Resource ID: Xenconfig_clone
Type: Filesystem
Select Clone
Clone_max: 2
Clone_node_max: 1
Clone or Master/Slave ID: Xenconfig_cloneset
```

Parameters:

```
Device: /dev/disk/by-id/<your-disk-id>
Directory /etc/xen/vm
Fstype: ocfs2
```

7. Now click *Add* to add the Cloneset. It will be added to the cluster now, but its status will be “not running”. That's fine for now.

8. Click the *Xenconfig_cloneset* that you have just created and add the following attributes:

Globally_unique: false

Notify: true

9. Select the first Xenconfig_clone and add the following operations:
Monitor; interval: 20, timeout: 60, prereq: nothing
Stop; timeout, 60
10. Now repeat steps 6-9 to create a cluster resource to store Xen data. Make sure that it is mounted on **/var/lib/xen/images**, for the rest you can use the exact same properties as the ones in steps 6-9 above.
11. Before going on, make sure that the OCFS2-related services are started automatically on your servers; you can accomplish this by using the **insserv o2cb** and **insserv ocfs2** commands on both servers. Next, make sure they are started before you activate the resources by using the commands **rco2cb start** and **rcofs2 start**.
12. Now it's time for the moment of truth: right-click on both clonesets and select *Start* to run them. To verify that it works, use the mount command on both servers. You should see that the volumes are available.

If you would like to use XML files as input, instead of configuring from the GUI, I recommend you to create an input file for each of the resources. For example, first create a file with the name **xenconfigcloneset.xml** and give it the following contents:

```
<clone id="xenconfigcloneset" notify="true" globally_unique="false" notify="true">
<instance_attributes id="xenconfigcloneset">
<attributes>
<nvpair id="xenconfigcloneset-01" name="clone_node_max" value="1"/>
<nvpair id="xenconfigcloneset-02" name="target_role" value="started"/>
</attributes>
</instance_attributes>
<primitive id="xenconfigclone" class="ocf" type="Filesystem" provider="heartbeat">
<operations>
<op name="monitor" interval="20s" timeout="60s" prereq="nothing" id="xenconfigclone-op-01"/>
<op name="stop" timeout="60s"/>
</operations>
<instance_attributes id="configstoreclone">
<attributes>
<nvpair id="xenconfigclone-01" name="device" value="/dev/sdb"/>
<nvpair id="xenconfigclone-02" name="directory" value="/etc/xen/vm"/>
<nvpair id="xenconfigclone-03" name="fstype" value="ocfs2"/>
</attributes>
</instance_attributes>
</primitive>
</clone>
```

Next, create a file with the name **xendatacloneset.xml** and give it the following contents:

```
<clone id="xendatacloneset" notify="true" globally_unique="false" notify="true">
<instance_attributes id="xendatacloneset">
<attributes>
```

```

<nvpair id="xendatacloneset-01" name="clone_node_max" value="1"/>
<nvpair id="xendatacloneset-02" name="target_role" value="started"/>
</attributes>
</instance_attributes>
<primitive id="xendataclone" class="ocf" type="Filesystem" provider="heartbeat">
<operations>
<op name="monitor" interval="20s" timeout="60s" prereq="nothing" id="xendataclone-op-01"/>
<op name="stop" timeout="60s"/>
</operations>
<instance_attributes id="datastoreclone">
<attributes>
<nvpair id="xendataclone-01" name="device" value="/dev/sdc"/>
<nvpair id="xendataclone-02" name="directory" value="/var/lib/xen/images"/>
<nvpair id="xendataclone-03" name="fstype" value="ocfs2"/>
</attributes>
</instance_attributes>
</primitive>
</clone>

```

Clicking or Typing?

For some people it is religion, in Heartbeat it is science. There are two ways of getting the information in your cluster, by clicking in the GUI or by creating your own XML files and inserting them with the cibadmin command. In the end, what you are going to use is probably a matter of taste. There is however one important advantage of creating your own XML files: it is so easy to re-use them. When working with Heartbeat, you will find that it is relatively easy to make errors. If you have made an error in complex resources, it may be a lot of work to recreate them by using the GUI. If on the contrary you have these resources defined in XML, it is very easy to remove them with the `crm_resource` command, and to apply them again, using `cibadmin -C`. Unfortunately, creating these XML files is not really the easiest job to do. To help you make it as easy as possible, the recently started Heartbeat Education project (<http://www.linux-ha.org/Education>) has made it one of its goals to put as much resource scripts as possible on the Heartbeat website.

As you can see, the XML files are almost the same. The only difference is in the device specification. Where in the GUI we have used the disk-ID as in `/dev/disk/by-id`, in the XML files I have pointed directly to the disk names `sdb` and `sdc` as they exist on the cluster nodes. That is something that you can do if you are sure that the disk names are the same on both nodes. If you are not, use the `disk/by-id` names instead.

Creating the Xen Cluster Resource

Now when the OCFS2 file systems are in place, it is time to create the Xen cluster resources. To do this, first install the Xen virtual machine on one of the servers, using the `virtmanager` utility from YaST. While installing the virtual machine, make sure that the disk image files are written to `/var/lib/xen/images` and that the disk configuration files are in `/etc/xen/vm` because this is where you have configured the shared OCFS2 volumes. After installing them, you can proceed to create the Xen

Cluster Resource. In the next example, we'll create a cluster resource for the Xen virtual machine named "lor".

1. From hb_gui, create a new cluster resource. When asked what type to use, enter "native".
2. Enter the following properties for the cluster resource:

Resource ID: lor

Type: Xen

Parameters:

xmfile: /etc/xen/vm/lor

If you prefer using an XML-file as input, create a file with for example the name **XenResource.xml** and give it the following contents:

```
<primitive id="lor" class="ocf" type="Xen" provider="heartbeat">
  <operations>
    <op name="monitor" interval="10s" timeout="60s" prereq="nothing" id="xen-op-01"/>
  </operations>
  <instance_attributes id="lor_instance_attrs">
    <attributes>
      <nvpair id="lor_target_role" name="target_role" value="started"/>
      <nvpair id="a97441a8-1ca7-4f67-b2e9-9bf63e17b037" name="xmfile" value="/etc/xen/vm/lor"/>
    </attributes>
  </instance_attributes>
</primitive>
```

To put the XML file in your cluster, use **cibadmin -C -o resources -x XenResource.xml**.

Now that the Xen resource has been created, you must create an ordering constraint as well. This ensures that the Xen resource is brought up only after the OCFS2 volumes. As you can understand, this is a very important requirement because the Xen virtual machine gets its information from the OCFS2 file system; it simply wouldn't run if you inverse the load order.

1. From the hb_gui, create a new resource and select the order type. We need two ordering constraints, one for the Xenconfig filesystem resource and one for Xendata resource. These are the properties you need for them:

ID: lororderconstraints-01

From: lor

Type: after

To: xenconfigcloneset

ID: lororderconstraints-02

From: lor

Type: after

To: xendatacloneset

If you prefer inserting your own XML code instead of working from the GUI, the XML file should have the following contents:

© Sander van Vugt, August 2007; mail@sandervanvugt.com; www.sandervanvugt.com

Redistribution of this document is free, as long as this footer is distributed as well.

```
<rsc_order id="lororderconstraints-01" from="lor" type="after" to="xenconfigcloneset"/>  
<rsc_order id="lororderconstraints-02" from="lor" type="after" to="xendatacloneset"/>
```

To put the XML file in your cluster, use the command **cibadmin -C -o constraints -x lorconstraints**.

2. Now you can highlight the Xen resource, right-click it and next select *Start*. This should activate the virtual machine as a resource in the cluster.

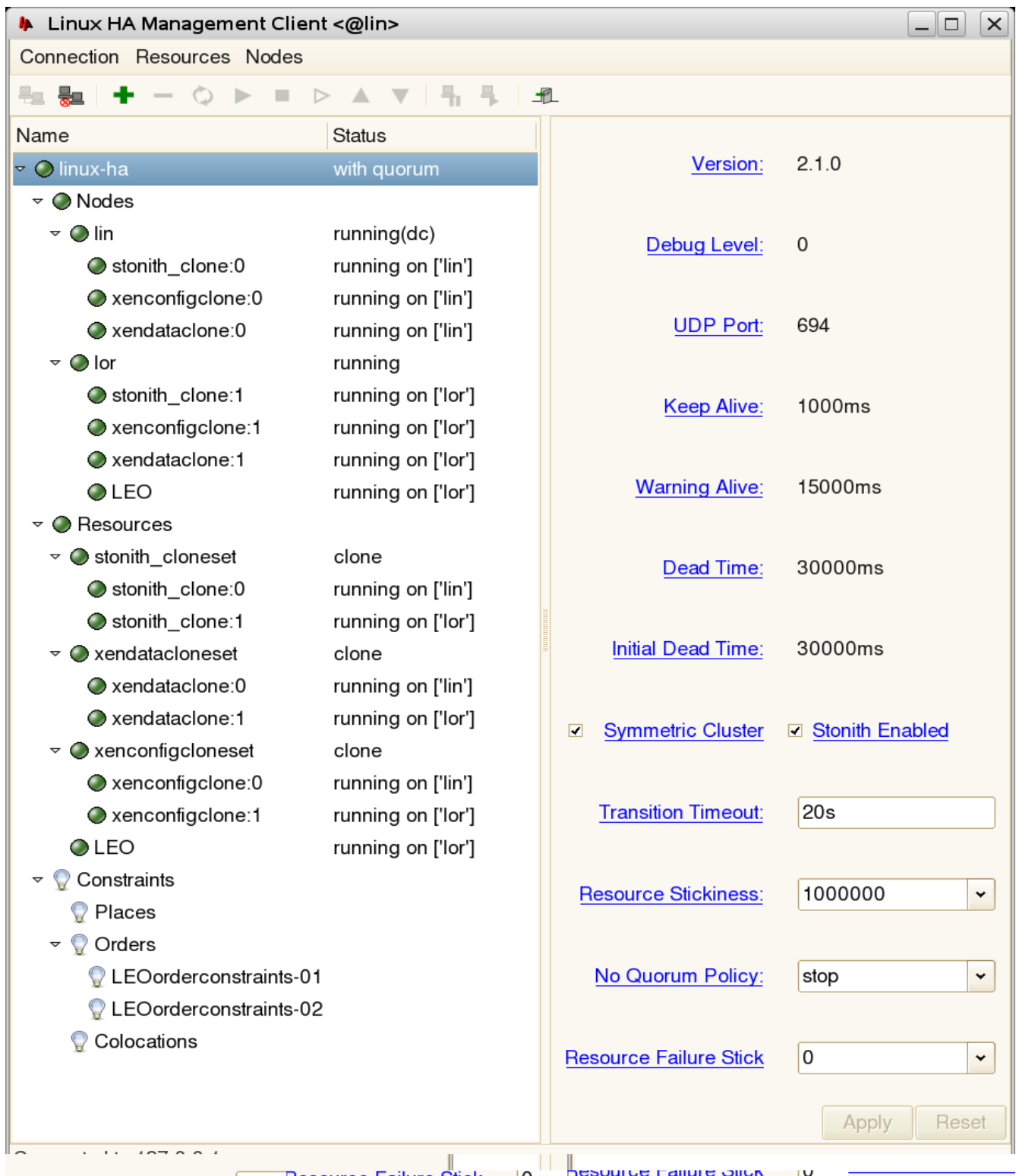


Figure 7: This is what success looks like.

Everything is configured now, and if it's all OK, the `hb_gui` should show you an overview like in figure 7. If you successfully arrived here, it's time to do some testing now. The best thing to do? Have a look at what machine the virtual machine is currently running on. On the other cluster node, just pull the plug. This should fail over and restart the virtual machine on the other cluster node. Congratulations, you now have configured your Xen high availability solution.

Summary

In this article you have read how Heartbeat can be used on SUSE Linux Enterprise Server to create a high availability solution. In this high-availability solution, we have used Xen virtual machines on top of an OCFS2 cluster safe file system.

Sander van Vugt