

Installation Guide for SLES10 and RHEL5

This installation guide describes how to set up a test HA cluster for the OCF IDS resource agent (RA) for Linux-HA aka Heartbeat on Suse Linux Enterprise Server 10 (SLES10) and Red Hat Enterprise Linux 5 (RHEL5). Though this guide only describes the steps on SLES10 and RHEL5, this should also work – with slight modifications – on other Linux distributions as well.

The machines used for setting up the test cluster while writing this installation guide did not have an internet connection due to company internal firewall rules. Therefore all packages needed are either installed from the DVDs or copied over from a machine that has an internet connection. So if your machines have an internet connection you can have the SLES10 or RHEL5 package managers resolve dependencies automatically for you and skip some of the steps of this guide.

For the described test setup four machines in total are used. Three machines serve as cluster nodes for Heartbeat and one machine functions as a time server via the Network Time Protocol (NTP) and as a shared storage server via the Network File System (NFS). The NTP server is needed to keep the system time synchronous between the cluster nodes. The NFS serves as a shared storage and will be mounted only on the node which currently holds the cluster resources. Of course, any other type of shared storage can be used, but this guide does not cover those.

Primarily, this guide describes the steps as they are performed on SLES10. If a step is different on RHEL5, this will be pointed out separately in a text box with red background color. Steps specific to SLES10 are presented in a text box with green background color. Normal text and text boxes with no background color refer to SLES10 and RHEL5 or maybe even to any Linux distribution in general (this assumption is not verified).

The machines in the test cluster running on SLES10 will have the following hostnames and IP addresses:

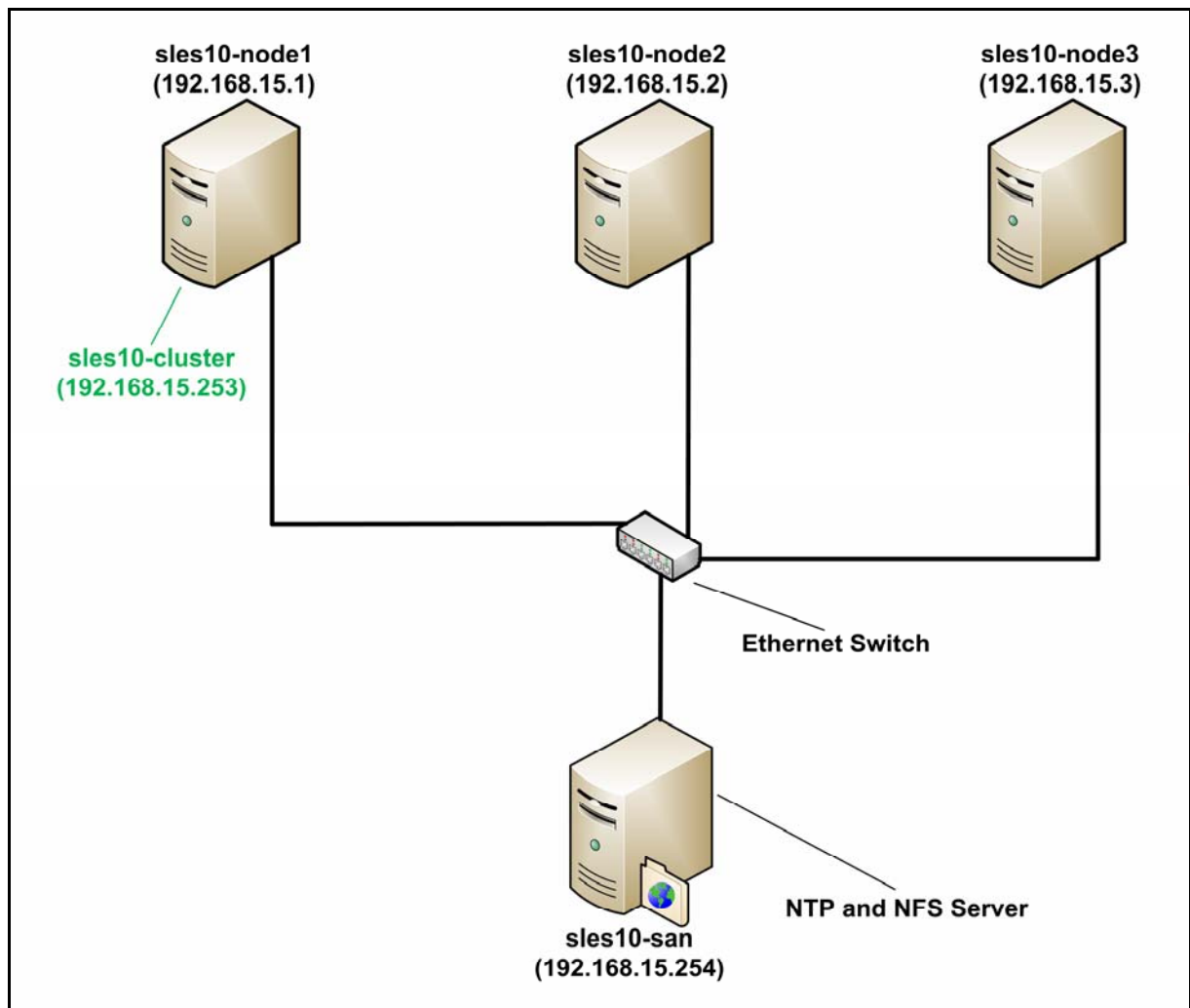
- First node: **sles10-node1, 192.168.15.1 on eth0**
- Second node: **sles10-node2, 192.168.15.2 on eth0**
- Third node: **sles10-node3, 192.168.15.3 on eth0**
- NTP and NFS server: **sles10-san, 192.168.15.254 on eth0**
- Virtual cluster IP address: **sles10-cluster, 192.168.15.253 on eth0**

The machines in the test cluster running on RHEL5 have the following hostnames and IP addresses:

- First node: **rhel5-node1, 192.168.15.1 on eth0**
- Second node: **rhel5-node2, 192.168.15.2 on eth0**
- Third node: **rhel5-node3, 192.168.15.3 on eth0**
- NTP and NFS server: **rhel5-san, 192.168.15.254 on eth0**
- Virtual cluster IP address: **rhel5-cluster, 192.168.15.253 on eth0**

The netmask is: **255.255.255.0**

The following figure illustrates how the setup looks like on SLES10 when *sles10-node1* is the current owner of the virtual cluster IP address:



This installation guide assumes that you have an appropriate copy (and license) of SLES10 and / or RHEL5 and IDS 11.

It is also assumed that all four machines have a working Ethernet network interface called eth0 and are connected to each other via an Ethernet switch and can communicate with each other. Other kinds of networking hardware are not covered here, as well as firewall or even security issues in general. Please search the Internet on these subjects if your setup differs in these points.

Neither the author of this installation guide nor IBM gives any warranty for this guide to work. Neither the author nor IBM is liable for defects caused by following the steps of this guide. Use this guide at your own risk!

Installing the Base System

Installing the base system is the first step. It will not be pointed out here in detail as the vendors of SLES10 and RHEL5 already have documented this very well. For help on installing SLES10 see the SLES10 product website of Novell at <http://www.novell.com/linux/>. For RHEL5 see <http://www.redhat.com/rhel/>.

- This guide assumes that you have installed SLES10 or RHEL5 with the X graphical user interface and the appropriate graphical system configuration utilities shipped with SLES10, respectively RHEL5.
- It is also assumed that during setup on the three machines to serve as cluster nodes a single partition formatted with the ext3 file system is created. On the machine the NTP and NFS servers run on, two partitions are needed: the first partition for the base system (mountpoint: `/`) and the second one for the shared storage (mountpoint: `/san`), both formatted with ext3. Of course, instead of ext3 any other file system could be used. The ext3 file system is used here for performance reasons of IDS.
- Configure the appropriate IP address and hostname (see listing above) for each machine. Be sure to use a static IP setup and to deactivate DHCP!
- The firewall is disabled during setup (if possible) otherwise it is assumed that you disable it yourself after the installation process. As mentioned before, this guide does not cover firewall or even security issues in general.
- Basic user authentication (via `/etc/passwd`) is chosen.
- Please be sure to install the Secure Shell (SSH) server and client if not done automatically as part of the base installation.
- **Additional Notes on RHEL5 setup:**

- It is recommended to deactivate the installation of the suites “Clustering” and “Storage Clustering” as they might interfere with Heartbeat later on. They’re not needed anyway!
- Also disable SELinux, besides disabling the firewall as mentioned above.
- For reasons of simplicity and performance, disable Kdump as well.
- Do not enable NTP yet! This will be configured later on in this guide.

Further Network Configuration

- Add all hostnames and IP addresses to the file `/etc/hosts` on each machine in order to avoid the need for a DNS server. Verify the network setup via the ping command and logging on to all of the machines via SSH.
- If the above works, configure each of the machines to allow SSH login as user `root` from any of the other machines without supplying a password. This makes things much easier while setting up the cluster and can be undone once everything works. A short description on how to do this follows in the text box below.

SSH login without supplying a password:

Perform all the following steps as user `root`:

1. On the client run `ssh-keygen -t rsa` and remember where it stores the private key file (`id_rsa`) and the public key file (`id_rsa.pub`). Do not enter anything when asked for a passphrase; just hit the Enter key twice!
2. Copy the public key file (`id_rsa.pub`) from the client to the server by executing `scp id_rsa.pub root@<server>:/root/` on the client. Replace the '`<server>`' in the command by the appropriate hostname!
3. On the server, append the client's public key to the list of authorized keys by running `cat id_rsa.pub >> /root/.ssh/authorized_keys`.
4. If not done already, perform a SSH login from the client to the server in order to accept the server's fingerprint.
5. From now on you are able to login as user `root` from the client to the server via SSH.

The machine which will allow SSH login without supplying a password is referred to as 'server'. The machine that wants to login without supplying a password is referred to as 'client'.

Setting up the NTP and NFS Server

The first thing to do is to set up the machine (*sles10-san*, respectively *rhel5-san*) offering the NTP and NFS server and make sure they function correctly and are accessible by the three machines serving as cluster nodes.

SLES10 – Configuring the NTP server on host *sles10-san*:

1. Open `yast2` and click on “Network Services” in the left bar and then click on “NTP client” in order to open the NTP configuration utility.
2. Tell the NTP daemon to automatically run during boot.
3. Click on “Complex Configuration”.
4. Be sure that “Configure NTP Daemon via DHCP” is not enabled! Then select the synchronization type “Undisciplined Local Clock (LOCAL)” and click on “Edit”.
5. Click on “Driver Calibration”.
6. Set the value of “Stratum” to a value of “5” and click on “Next”.
7. Click on “Ok” and then on “Finish”.
8. Reboot system (just in case).
9. After the reboot, verify that the stratum of the local time device has a value of “5” by executing the following command: `ntpq -p`.

SLES10 – Configuring the NTP client on the cluster nodes:

1. Open `yast2` and click on “Network Services” in the left bar and then click on “NTP client” in order to open the NTP configuration utility.
2. Tell the NTP daemon to automatically run during boot.
3. Click on “Select” in the section “NTP Server Configuration” and select the point “Local NTP Server”.
4. Enter the hostname “`sles10-san`” in the field “Address”. Clicking on “Test” should tell you that the server is reachable. If not, please recheck the NTP configuration on host `sles10-san` above.
5. Click on “Ok” in order to get back to the main window of the NTP client configuration. Then click on “Finish” to complete your changes.
6. Restart the NTP server in order for the changes to take effect by running the following command in a shell: `/etc/init.d/ntp restart`.
7. Verify that the host `sles10-san` has a lower stratum than the local time device by executing the following command: `ntpq -p`. Using the `date` command on both systems can also be used to verify that the two systems clocks are in sync. Be sure to repeat this process on all nodes.
8. Sometimes the NTP configuration utility provided by SLES10 does not add the NTP server correctly. In this case you can also add the NTP server `sles10-san` manually by the command `echo "server sles10-san" >> /etc/ntp.conf`. After that, be sure to repeat steps 6 and 7.

RHEL5 – Configuring the NTP server on host *rhel5-san*:

1. Edit `/etc/ntp.conf` and replace “fudge 127.127.1.0 stratum 10” by “fudge 127.127.1.0 stratum 5”. Save your changes!
2. Open the “Service Configuration” tool via the menu or via the `serviceconf` command.
3. In the section “Background Services”, search for the entry “ntpd” and enable it. Then select it and click on the button labeled “Start” in order to start the NTP server daemon.
4. Click on the button labeled “Save” in order to save the changes.
5. Exit the “Service Configuration” tool.

RHEL5 – Configuring the NTP client on the cluster nodes:

1. Right-Click on the clock in the menubar and select the menu entry “Adjust Date&Time”.
2. You are prompted for the root password, enter it.
3. A window labeled “Date/Time Properties” opens. Select the section “Network Time Protocol”.
4. Enable the option “Enable Network Time Protocol”.
5. The section “NTP Servers” should now be accessible. Delete all servers listed in it.
6. Add a new server named “rhel5-san”.
7. Click on “Ok”.
8. Just to be sure the changes are really applied run `/etc/init.d/ntpd restart`.
9. Verify that the host *rhel5-san* has a lower stratum than the local time device by executing the following command: `ntpq -p`. Using the `date` command on both systems can also be used to verify that the two systems clocks are in sync. Be sure to repeat this process on all nodes.

SLES10 – Configuring the NFS server on host *sles10-san*:

1. Open yast2 and click on “Network Services” in the left bar and then click on “NFS Server” in order to open the NFS server configuration utility.
2. In the section “NFS Server”, select the option “Start” and then click on “Next” in the lower right corner of the window.
3. Click on “Add Directory” and select the directory where previously the second partition was mounted to for the NFS server (“/san”) and then click on “Ok”.
4. A small window options and prompts you to specify the options for this exported directory. Be sure to enter “*” in the field “Host Wild Card”. In the field “Options” enter “rw, sync, no_root_squash” and click “Ok”.
5. Click on “Finish”.

RHEL5 – Configuring the NFS server on the host *rhel5-san*:

1. Open the “Service Configuration” tool via the menu or via the `serviceconf` command.
2. In the section “Background Services”, search for the entry “nfs” and enable it. Then select it and click on the button labeled “Start” in order to start the NFS server daemon.
3. Click on the button labeled “Save” in order to save the changes.
4. Exit the “Service Configuration” tool.
5. Create the directory `/san` via `mkdir /san`. It will hold the shared data of the NFS share.
6. Open the “NFS Server Configuration” tool via the menu or via the `system-config-nfs` command. Depending on how RHEL5 was installed, you will have to install this tool before proceeding.
7. In the window “NFS Server Configuration”, click on the button labeled “Add”. A new window labeled “Add NFS Share” opens.
8. In the section “Basic” of the new window, enter “`/san`” in the field named “Directory”. Also enter “`*`” for the field “Host(s)”. In addition, enable the option “Read / Write”.
9. In the section “User Access”, enable the option “Treat remote root user as local root”.
10. Click on “Ok” in order to close the window “Add NFS Share”.
11. Close the window “NFS Server Configuration”.
12. In order to be sure the changes take effect, run `/etc/init.d/nfs restart` in order to restart the NFS server.

Verify the NFS server setup on each cluster node:

Note: **On RHEL5, the hostnames have to be replaced accordingly.**

1. Create the mountpoint for the remote storage via `mkdir /mnt/san`.
2. Mount the NFS share via `mount sles10-san:/san /mnt/san/`.
3. Verify the NFS share is mounted correctly by simply invoking `mount`.
4. Verify that the node has write permissions on the NFS share by creating a test file: `touch /mnt/san/test-from-sles10-node1; ls /mnt/san`.
5. If all went well, you should have been able to create a test file in the NFS share from each of the three nodes.
6. Before proceeding with the guide, be sure to unmount the NFS share on all nodes by running `umount /mnt/san` on each node.

Installing Informix Dynamic Server (IDS)

This installation guide assumes you have received IDS11 for Linux as a simple tar file. If you received it in different format or as an already extracted folder structure, it is possible that you can skip some of the first steps here. Repeat the following steps on each of the cluster nodes.

Note: On RHEL5, the hostnames have to be replaced accordingly.

1. Create the user and group *informix* and make sure that the userid and groupid are the same on each of the nodes (i.e. 1001) and that user *informix* is a member of the group *informix*.
2. Create directory where IDS should be installed to and change the owner and group of it to be "informix": `mkdir /informix && chown informix:informix /informix`.
3. Copy the IDS11 tar file to */informix* and unpack it there. After unpacking the tar file can be removed, if desired.
4. Create a directory for the IDS log files and create the log files and change ownership to user and group *informix*: `cd /informix && mkdir logs && touch logs/console.log && touch logs/online.log && chown -R informix:informix logs/`.
5. **This step needs to be performed on only one of the nodes:** Mount the NFS share via `mount sles10-san:/san /mnt/san/`. Then create a directory where IDS should store its databases and change ownership to user and group *informix*: `mkdir /mnt/san/informix-data/ && chown informix:informix /mnt/san/informix-data/`. Create the root chunk for IDS, change ownership to user and group *informix* and give it the file permissions "660": `touch /mnt/san/informix-data/online_root && chown informix:informix /mnt/san/informix-data/online_root && chmod 660 /mnt/san/informix-data/online_root`.
6. Adapt the IDS setup configuration file at */informix/server.ini* by replacing `-G licenseAccepted="false"` by `-G licenseAccepted="true"` and `-p installLocation="/usr/informix"` by `-p installLocation="/informix"`.
7. Start the IDS setup as user *root* via `cd /informix && ./installserver -options "server.ini" -silent`.
8. Copy the standard configuration file in */informix/etc/* and give it a name that refers to the IDS instance that should run as a resource in the cluster (here: "ids1"). This file has to be owned by user and group *informix* with permissions

set to "660". The following commands do all this: `cd /informix/etc && cp onconfig.std onconfig.ids1 && chown informix:informix onconfig.ids1 && chmod 660 onconfig.ids1.`

9. Adapt the file *onconfig.ids1*:

- Change "ROOTPATH" to "/mnt/san/informix-data/online_root"
- Change "ROOTSIZE" to "100000"
- Change "PHYSFILE" to "20000"
- Change "LOGSIZE" to "20000"
- Change "MSGPATH" to "/informix/logs/online.log"
- Also change all other paths from "/usr/informix" to "/informix"
- Change "DUMPPDIR" to "/tmp"
- Change "LTAPEDEV" to "/dev/null"
- Change "DBSERVERNAME" to "ids1"

10. Create the file *sqlhosts* in */informix/etc/* with file permissions set to "660" and change ownership to user and group *informix*: `cd /informix/etc/ && echo "ids1 onsoctcp sles10-cluster ids1" > sqlhosts && chown informix:informix sqlhosts && chmod 660 sqlhosts`. Just to remind you: if you are on RHEL5 the hostname in this command would be "rhel5-cluster"! This tells IDS to create a socket and listen for database connections on a TCP port.

11. Now add (append) a service name and TCP port for IDS in the */etc/services* file: `echo "ids1 80321/tcp" >> /etc/services`. The port "80321" could be replaced by any other port number here. Please note the ">>" instead of simply ">" which otherwise would overwrite the file instead of appending to it!

12. Before being able to use IDS correctly, you have to set several shell environment variables. Best practice is to store them in a shell script and executing that script before using IDS. Here's the script how it looks like in this setup constellation:

```
#!/bin/sh

export INFORMIXDIR="/informix"
export INFORMIXSERVER="ids1"
export ONCONFIG="onconfig.${INFORMIXSERVER}"
export PATH="${INFORMIXDIR}/bin:${PATH}"
export LD_LIBRARY_PATH="${INFORMIXDIR}/lib:${INFORMIXDIR}/lib/esql"
export LD_LIBRARY_PATH
```

13. Place the above shell script in the new file */informix/env-ids1.sh* and give it file permissions "755". Now you just have to run `./informix/env-ids1.sh` from anywhere and you are able to use IDS.

14. Mount the NFS share, if not done so already: `mount sles10-san:/san /mnt/san/.`

15. As Heartbeat is not installed and configured yet to start the virtual IP address for the HA cluster, we have to assign it here manually: `ifconfig eth0:1 192.168.15.253`. If you forget to perform this step, IDS will not start and terminate with an error message like “Network driver cannot bind a name to the port. System error = 99.”.
16. **This step needs to be performed on only one of the nodes:** Now let IDS initialize the root dbspace by executing `oninit -i`. If you initialized IDS already on a previous node, just run `oninit` here instead in order to check if IDS starts successfully.
17. Verify the state of IDS with the command `onstat -`. This should print something like the following: “IBM Informix Dynamic Server Version 11.10.UB7 -- On-Line – Up 00:01:36”. If this is not the case, please check the log file at `/informix/logs/online.log` for hints on resolving the issue.
18. **This step needs to be done on only one of the nodes:** The command `dbaccessdemo7` creates a demo database with sample tables and data in it.
19. Once the demo database is created, run the following command in order to pass a sample SQL query to the previously created demo database: `echo "SELECT * FROM items;" | dbaccess stores_demo -`.
20. In order to stop IDS again, run the following command: `onmode -kuy`. You can verify this via the `onstat -` command. The output should look something like: “shared memory not initialized for INFORMIXSERVER ‘ids1’”.
21. Before proceeding to install IDS on the other nodes or proceeding with the installation guide, unmount the NFS share via `umount /mnt/san/` and drop the virtual IP via `ifconfig eth0:1 down`.

Installing Linux-HA aka Heartbeat

Before you can install Heartbeat, you need to obtain the latest official release from the Linux-HA web site. Here is a direct link to the official Linux-HA download section: <http://linux-ha.org/download/index.html>. Use these packages on SLES10.

Note that the official RPMs from the Linux-HA website are built against Suse and will most likely not work on RHEL5!

Therefore use the packages provided by CentOS for installing Heartbeat on RHEL5: <http://mirror.centos.org/centos/5/extras/i386/RPMS/>.

There are other resources for obtaining Linux-HA, but this installation guide will only cover the official packages from the Linux-HA website. This could mean not getting any commercial support for these packages. Please refer to your contact person on these topics. The Heartbeat packages shipped with SLES10, for instance, should work though. Nevertheless, it is possible that they contain bugs which have been resolved in the official release. It is up to you which packages you use...

As SLES10 and RHEL5 are based on the Red Hat Package Manager (RPM) in order to maintain the installed packages on the system only the RPM installation is covered here. Other package formats (such as .deb) should work analogous though. So this guide assumes you download the RPM packages from the URL stated above.

When this guide was written, the newest release was version 2.1.2. Therefore this guide will refer to Heartbeat 2.1.2 when using the term Heartbeat. This guide should work for later releases as well, though.

Install Heartbeat on all cluster nodes by repeating the following steps on each node:

1. Create the directory `/root/heartbeat-2.1.2/` and copy the previously downloaded RPMs into it. (**Remember to use the CentOS RPMs for RHEL5!**)
2. In `/root` run `rpm -i heartbeat-2.1.2/heartbeat-*` in order to install the Heartbeat packages which are in this case: `heartbeat-2.1.2-2.i586.rpm`, `heartbeat-gui-2.1.2-2.i586.rpm`, `heartbeat-pils-2.1.2-i586.rpm` and `heartbeat-stonith-2.1.2-i586.rpm`
- 3.

On a freshly installed SLES10 this produces the following output:

```
sles10-node1:~ # rpm -i heartbeat-2.1.2/heartbeat-*
warning: heartbeat-2.1.2/heartbeat-2.1.2-2.i586.rpm: Header V3 DSA
signature: NOKEY, key ID 4dab7996
error: Failed dependencies:
libnet.so.0 is needed by heartbeat-2.1.2-2.i586
libopenhpi.so.2 is needed by heartbeat-stonith-2.1.2-2.i586
sles10-node1:~ #
```

This means you have to install the packages “libnet” and “openhpi” before proceeding. When installing these packages via Yast2, the dependencies of the two needed packages should be resolved automatically.

On a freshly installed RHEL5 this produces the following output:

```
[root@rhel5-node2 ~]# rpm -i heartbeat-2.1.2-RHEL5/*
warning: heartbeat-2.1.2-RHEL5/heartbeat-2.1.2-2.el5.centos.i386.rpm:
Header V3 DSA signature: NOKEY, key ID e8562897
error: Failed dependencies:
    libsensors.so.3 is needed by heartbeat-2.1.2-
2.el5.centos.i386
    swig is needed by heartbeat-gui-2.1.2-2.el5.centos.i386
    libopenhpi.so.2 is needed by heartbeat-stonith-2.1.2-
2.el5.centos.i386
[root@rhel5-node2 ~]#
```

Please install the missing packages “lm_sensors”, “swig” and “openhpi” and resolve potential further dependencies before proceeding with the installation guide.

4. After resolving the dependencies for the Heartbeat packages in step 3, retry installing Heartbeat by running `rpm -i heartbeat-2.1.2/heartbeat-*` in `/root`.

Basic Heartbeat Configuration

Once Heartbeat is successfully installed, it needs to be configured. Best practice is to configure Heartbeat on one of the nodes and then copy the configuration files over to the other nodes. This saves time and avoids errors. This guide will not use the GUI that is shipped with Heartbeat as it does not yet fully support all features of Heartbeat Version 2 and it still has some bugs. Furthermore, configuring Heartbeat and its resources manually by using the command line tools and passing XML files to it has the advantage of getting a better understanding how Heartbeat works and having the configuration parts stored in reusable XML files.

If you want to use the GUI though, do not forget to set a password for the user *hacluster* which has been created during the installation process of Heartbeat. In addition, the GUI depends on the “python-xml” package. Furthermore, instead of reading this section you could also watch the screencast on installing and configuring Heartbeat created by Alan Robertson. The screencast can be found at <http://linux-ha.org/Education/Newbie/InstallHeartbeatScreencast>.

The following steps describe how a basic Heartbeat configuration is done:

Note: **On RHEL5, the hostnames have to be replaced accordingly.**

1. In order to save time, you can copy and adapt two of the shipped sample configuration files. Just run these commands: `cp /usr/share/doc/heartbeat-2.1.2/authkeys /etc/ha.d/ && cp /usr/share/doc/heartbeat-2.1.2/authkeys /etc/ha.d/`. The *ha.cf* configuration file will be created manually as the shipped sample *ha.cf* file is way too big in order to describe the needed changes in an appropriate time here.
2. Edit */etc/logd.cf* and replace “#logfacility local7” by “logfacility daemon”. This tells the Heartbeat log daemon to forward its messages to the system logging daemon which (per default) writes its messages to */var/log/messages*. Now is a good time to copy this file over to the other nodes.
3. Edit */etc/ha.d/authkeys* and replace “#auth 1” by “auth 2” and “#2 sha1 HI!” by “2 sha1 This is a simple test cluster on Heartbeat! :)”. Of course, you can choose a different passphrase here! This tells Heartbeat to use the SHA1 algorithm for signing the heartbeat packets between the nodes. Be sure to

give the *authkeys* file the permissions “600”! Then, copy this file as well over to the other nodes and verify the file permissions on the other nodes again as they might have been modified during the copy process.

4. Create the file */etc/ha.d/ha.cf* with the following content:

```
use_logd yes
bcast eth0
node sles10-node1
node sles10-node2
node sles10-node3
ping sles10-san
crm yes
```

This tells Heartbeat to...

- enable the built-in logging daemon
- send heartbeat packets as broadcasts over the network interface *eth0*
- add three members to the cluster: *sles10-node1*, *sles10-node2* and *sles10-node3*
- use *sles10-san* as a ping node
- enable the Cluster Resource Manager (CRM)

It can occur that the node names on RHEL5 have to be specified including the domain name which was defined during the network setup at the beginning of this installation guide. In such a case you will probably see error messages that look something like the following:

```
Aug 10 15:06:19 rhel5-node1 heartbeat: [2712]: ERROR: Current node
[rhel5-node1.ibm.com] not in configuration!
Aug 10 15:06:19 rhel5-node1 heartbeat: [2712]: info: By default,
cluster nodes are named by `uname -n` and must be declared with a
'node' directive in the ha.cf file.
Aug 10 15:06:19 rhel5-node1 heartbeat: [2712]: info: See also:
http://linux-ha.org/ha.cf/NodeDirective
Aug 10 15:06:19 rhel5-node1 heartbeat: [2712]: ERROR: Configuration
error, heartbeat not started.
```

In this case, just append the domain “*ibm.com*” to the node names defined in the *ha.cf* file, for example: replace “*rhel5-node1*” by “*rhel5-node1.ibm.com*”.

→ Don't forget to copy the *ha.cf* file over to the other nodes as well!

Firing up the HA Cluster

- After installing Heartbeat and doing a basic configuration on all nodes, it is time to start Heartbeat by running `/etc/init.d/heartbeat/start` on all of the nodes.
- It is always a good idea to watch the log files via `tail -f /var/log/messages` on each of the nodes in order to keep track of possible difficulties.
- Via `crm_mon -i 5` the current status of the cluster, its nodes and its resources is displayed and refreshed every five seconds. If everything worked well until now, the `crm_mon` tool should print something like this:

```
Refresh in 3s...

=====
Last updated: Mon Aug  6 18:29:24 2007
Current DC: sles10-node1 (58745d70-c47e-40c4-bb4f-ae35f39eab78)
3 Nodes configured.
0 Resources configured.
=====

Node: sles10-node1 (58745d70-c47e-40c4-bb4f-ae35f39eab78): online
Node: sles10-node2 (1278e271-a3ec-4bae-a647-27bb2d3c42a3): online
Node: sles10-node3 (7daceb49-7fbd-4ef5-9183-66911ed4d3d5): online
```

Advanced Heartbeat Configuration

Now that the HA cluster, including all of its nodes, is functional and running it is time to configure a few resources the HA cluster should manage and fail over when needed. This guide will add the following resources to the HA cluster:

- A virtual IP address for the cluster
- A NFS share used as shared storage
- An IDS database server instance
- A pingd resource in order to check each node's connectivity status
- A meatware STONITH device that requests a manual reboot of a dead node

Thereby the virtual IP address, the NFS share and the IDS instance are summarized in a group. This makes things easier as the three depend on each other and, by default, Heartbeat will only run the resource group as one. So if one of the resources of the group fails, the group is moved to another node. Furthermore, the resources in a resource group are, by default, started in the same order as defined in the group. Defining the resource for the NFS share before the resource for the IDS instance keeps IDS from trying to access a NFS share not mounted yet!

Note: **On RHEL5, the hostnames have to be replaced accordingly.**

The above resource group definition is stored in a file called *ids_validation_cluster_group.xml* which has the following contents:

```
<group id="ids_validation_cluster_group" collocated="true" ordered="true">
  <instance_attributes id="ids_validation_cluster_instance_attrs">
    <attributes>
      <nvpair id="ids_validation_cluster_target_role" name="target_role"
value="started"/>
    </attributes>
  </instance_attributes>

  <!-- Filesystem-NFS -->
  <primitive id="cNFS" class="ocf" type="Filesystem" provider="heartbeat"
is_managed="true">
    <instance_attributes id="cNFS_instance_attrs">
      <attributes>
        <nvpair name="options" value="rw"/>
        <nvpair name="device" value="sles10-san:/san"/>
        <nvpair name="directory" value="/mnt/san"/>
      </attributes>
    </instance_attributes>
    <operations>
      <op name="monitor" interval="20" timeout="40" start_delay="10"/>
    </operations>
  </primitive>

  <!-- IPAddr2 -->
  <primitive class="ocf" type="IPAddr2" provider="heartbeat" is_managed="true" id="cIP">
    <instance_attributes id="cIP_instance_attrs">
      <attributes>
        <nvpair name="ip" value="192.168.15.253"/>
        <nvpair name="nic" value="eth0"/>
        <nvpair name="mac" value="auto"/>
        <nvpair name="cidr_netmask" value="24"/>
      </attributes>
    </instance_attributes>
    <operations>
      <op name="monitor" interval="10s" timeout="20s" start_delay="5s"/>
    </operations>
  </primitive>

  <!-- ids -->
  <primitive class="ocf" type="ids" provider="heartbeat" is_managed="true" id="cIDS">
    <instance_attributes id="cIDS_instance_attrs">
      <attributes>
        <nvpair name="informixdir" value="/informix"/>
        <nvpair name="informixserver" value="ids1"/>
        <nvpair name="onconfig" value="onconfig.ids1"/>
        <nvpair name="dbname" value="stores_demo"/>
        <nvpair name="sqltestquery" value="SELECT COUNT(*) FROM items;"/>
      </attributes>
    </instance_attributes>
    <operations>
      <op name="monitor" interval="10" timeout="30" start_delay="10"/>
    </operations>
  </primitive>

</group>
```

The pingd resource is configured by the file *pingd-resource.xml*:

```
<clone id="pingd">
  <instance_attributes id="pingd">
    <attributes>
      <nvpair id="pingd-clone_node_max" name="clone_node_max" value="1"/>
    </attributes>
  </instance_attributes>
  <primitive id="pingd-child" provider="heartbeat" class="ocf" type="pingd">
    <operations>
      <op id="pingd-child-monitor" name="monitor" interval="20s" timeout="40s"
prereq="nothing"/>
      <op id="pingd-child-start" name="start" prereq="nothing"/>
    </operations>
    <instance_attributes id="pingd_inst_attr">
      <attributes>
        <nvpair id="pingd-dampen" name="dampen" value="5s"/>
        <nvpair id="pingd-multiplier" name="multiplier" value="100"/>
      </attributes>
    </instance_attributes>
  </primitive>
</clone
```

In addition four resource location constraints are defined in order to guarantee that the resource group and the meatware STONITH device only runs on a node that can at least reach one of the ping nodes. If several nodes fulfill this requirement, the node which can reach the most ping nodes is chosen.

The first resource location constraint is set via the file *ids_validation_cluster_group-resource-location-pingd-never.xml*:

```
<rsc_location id="pingd:never" rsc="ids_validation_cluster_group">
  <rule id="pingd:never:rule" score="-INFINITY" boolean_op="or">
    <expression id="pingd:never:expr:undefined" attribute="pingd"
operation="not_defined"/>
    <expression id="pingd:never:expr:zero" attribute="pingd" operation="lte"
value="0"/>
  </rule>
</rsc_location>
```

The second resource location constraint is found in the file *ids_validation_cluster_group-resource-location-pingd-gt0.xml*:

```

<rsc_location id="pingd:connected" rsc="ids_validation_cluster_group">
  <rule id="pingd:connected:rule" score_attribute="pingd" boolean_op="and">
    <expression id="pingd:connected:expr:defined" attribute="pingd"
operation="defined" />
    <expression id="pingd:connected:expr:gt0" attribute="pingd" operation="gt"
value="0" />
  </rule>
</rsc_location>

```

The meatware STONITH resource is configured by the file *stonith-meatware-resource.xml*:

```

<clone id="stonith_meatware_clone">
  <instance_attributes>
    <attributes>
      <nvpair id="stonith_meatware-clone_node_max" name="clone_node_max" value="1"/>
    </attributes>
  </instance_attributes>
  <primitive id="stonith_meatware" class="stonith" type="meatware"
provider="heartbeat">
    <instance_attributes id="stonith_meatware_instance_attrs">
      <attributes>
        <nvpair id="stonith_meatware_hostlist" name="hostlist" value="sles10-node1
sles10-node2 sles10-node3" />
      </attributes>
    </instance_attributes>
    <operations>
      <op name="monitor" interval="40s" timeout="60s"/>
      <op name="start" timeout="60s"/>
    </operations>
  </primitive>
</clone>

```

The third resource location constraint is found in the file *stonith-meatware-resource-location-pingd-never.xml*:

```

<rsc_location id="meatware:pingd:never" rsc="stonith_meatware_clone">
  <rule id="meatware:pingd:never:rule" score="-INFINITY" boolean_op="or">
    <expression id="meatware:pingd:never:expr:undefined" attribute="pingd"
operation="not_defined"/>
    <expression id="meatware:pingd:never:expr:zero" attribute="pingd" operation="lte"
value="0"/>
  </rule>
</rsc_location>

```

The fourth resource location constraint is found in the file *stonith-meatware-resource-location-pingd-gt0.xml*:

```
<rsc_location id="meatware:pingd:connected" rsc="stonith_meatware_clone">
  <rule id="meatware:pingd:connected:rule" score_attribute="pingd" boolean_op="and">
    <expression id="meatware:pingd:connected:expr:defined" attribute="pingd"
operation="defined" />
    <expression id="meatware:pingd:connected:expr:gt0" attribute="pingd"
operation="gt" value="0" />
  </rule>
</rsc_location>
```

In order to add the previously described resources and constraints to the cluster perform the following steps, except for step 3, **only on one node** (changes to the Cluster Information Base (CIB) are replicated by Heartbeat automatically to the other nodes):

1. As we are going to use a STONITH device, we have to explicitly enable STONITH support in Heartbeat. This is done via the following command:
`crm attribute -n stonith-enabled -v true`. Forgetting to perform this step will lead to the configured STONITH devices not being invoked correctly and can cause a lot of frustration while analyzing the issue! So double check this step!
2. Create the files with the contents and names as described above and place them in `/root`.
3. **Before doing anything else, the IDS resource agent has to be installed on all nodes otherwise you will run into problems!** Though this step only has to be done if the IDS resource agent is not shipped with Heartbeat which is the case in release 2.1.2. You should have obtained two separate files for the IDS resource agent (RA) with this installation guide. If not, kindly ask for a copy on the Linux-HA mailing list (see <http://linux-ha.org/ContactUs>). In order to install the IDS RA, copy the wrapper script of the IDS RA to `/etc/ha.d/resource.d/` and copy the OCF script to `/usr/lib/ocf/resource.d/heartbeat/`. Be sure to give both files the file permissions "755", respectively the same rights as the other RAs have.
4. Change the current directory to `/root` where the files created in step 1 reside.
5. Add the resource group "ids_validation_cluster_group" via the following command to the cluster:
`cibadmin -C -o resources -x ids_validation_cluster_group.xml`
6. Add the pingd resource via the following command:
`cibadmin -C -o resources -x pingd-resource.xml`
7. Add the meatware STONITH resource via the following command:
`cibadmin -C -o resources -x stonith-meatware-resource.xml`

8. Add the first resource location constraint via the following command:
`cibadmin -C -o constraints -x ids_validation_cluster_group-resource-location-pingd-never.xml`
9. Add the second resource location constraint via the following command:
`cibadmin -C -o constraints -x ids_validation_cluster_group-resource-location-pingd-gt0.xml`
10. Add the third resource location constraint via the following command:
`cibadmin -C -o constraints -x stonith-meatware-resource-location-pingd-never.xml`
11. Add the fourth resource location constraint via the following command:
`cibadmin -C -o constraints -x stonith-meatware-resource-location-pingd-gt0.xml`
12. If all went well, the output of `crm_mon` should look something like this:

```
Refresh in 5s...

=====
Last updated: Tue Aug  7 16:06:03 2007
Current DC: sles10-node3 (7daceb49-7fbd-4ef5-9183-66911ed4d3d5)
3 Nodes configured.
3 Resources configured.
=====

Node: sles10-nodel1 (58745d70-c47e-40c4-bb4f-ae35f39eab78): online
Node: sles10-node2 (1278e271-a3ec-4bae-a647-27bb2d3c42a3): online
Node: sles10-node3 (7daceb49-7fbd-4ef5-9183-66911ed4d3d5): online

Resource Group: ids_validation_cluster_group
  cNFS      (heartbeat::ocf:Filesystem):   Started sles10-nodel1
  cIP (heartbeat::ocf:IPaddr2):       Started sles10-nodel1
  cIDS      (heartbeat::ocf:ids):       Started sles10-nodel1
Clone Set: pingd
  pingd-child:0      (heartbeat::ocf:pingd): Started sles10-nodel1
  pingd-child:1      (heartbeat::ocf:pingd): Started sles10-node3
  pingd-child:2      (heartbeat::ocf:pingd): Started sles10-node2
Clone Set: stonith_meatware_clone
  stonith_meatware:0 (stonith:meatware):   Started sles10-node2
  stonith_meatware:1 (stonith:meatware):   Started sles10-node3
  stonith_meatware:2 (stonith:meatware):   Started sles10-nodel1
```

Testing the HA Cluster Setup

In order to test the previously set up HA cluster, you can run a very common test scenario as described by the following steps:

Note: On RHEL5, the hostnames have to be replaced accordingly.

1. Simulate a broken network interface on one of the nodes, say *sles10-node1*, by running the following command on *sles10-node1*: `ifconfig eth0 down`
2. The other nodes will notice the absence of *sles10-node1* and declare it as dead. The node *sles10-node1* on the other hand thinks that the other nodes are dead and it is the only one left in the cluster. This is a classic split-brain situation that needs to be resolved before deciding what to do with the cluster resources. The node on which the meatware STONITH resource is running will print the following message in the log file (*/var/log/messages*):

```
Aug  7 17:09:11 sles10-node3 tengine: [3660]: info: te_fence_node:
Executing reboot fencing operation (39) on sles10-node1
(timeout=30000)
Aug  7 17:09:11 sles10-node3 stonithd: [3115]: info: client tengine
[pid: 3660] want a STONITH operation RESET to node sles10-node1.
Aug  7 17:09:11 sles10-node3 stonithd: [3115]: info:
stonith_operate_locally::2532: sending fencing op (RESET) for sles10-
node1 to device meatware (rsc_id=stonith_meatware:1, pid=8528)
Aug  7 17:09:11 sles10-node3 stonithd: [8528]: CRIT: OPERATOR
INTERVENTION REQUIRED to reset sles10-node1.
Aug  7 17:09:11 sles10-node3 tengine: [3660]: info: te_pseudo_action:
Pseudo action 34 fired and confirmed
Aug  7 17:09:11 sles10-node3 stonithd: [8528]: CRIT: Run "meatclient
-c sles10-node1" AFTER power-cycling the machine.
Aug  7 17:09:11 sles10-node3 tengine: [3660]: info: te_pseudo_action:
Pseudo action 38 fired and confirmed
```

3. Manually reboot *sles10-node1* via `reboot`.
4. Inform the node in which's log file the reboot of *sles10-node1* was requested via `meatclient -c sles10-node1`.
5. The resources will be moved to one of the two remaining nodes and the output of `crm_mon` should now look something like this:

```
Refresh in 5s...

=====
Last updated: Tue Aug  7 17:12:34 2007
Current DC: sles10-node3 (7daceb49-7fbd-4ef5-9183-66911ed4d3d5)
3 Nodes configured.
3 Resources configured.
=====

Node: sles10-node1 (58745d70-c47e-40c4-bb4f-ae35f39eab78): OFFLINE
Node: sles10-node2 (1278e271-a3ec-4bae-a647-27bb2d3c42a3): online
Node: sles10-node3 (7daceb49-7fbd-4ef5-9183-66911ed4d3d5): online

Resource Group: ids_validation_cluster_group
  cNFS      (heartbeat::ocf:Filesystem):    Started sles10-node2
  cIP (heartbeat::ocf:IPaddr2):          Started sles10-node2
  cIDS      (heartbeat::ocf:ids):          Started sles10-node2
Clone Set: pingd
  pingd-child:0      (heartbeat::ocf:pingd): Stopped
  pingd-child:1      (heartbeat::ocf:pingd): Started sles10-node3
  pingd-child:2      (heartbeat::ocf:pingd): Started sles10-node2
Clone Set: stonith_meatware_clone
  stonith_meatware:0 (stonith:meatware):    Started sles10-node2
  stonith_meatware:1 (stonith:meatware):    Started sles10-node3
  stonith_meatware:2 (stonith:meatware):    Stopped
```

6. Once *sles10-node1* is done with rebooting, start Heartbeat again via `/etc/init.d/heartbeat start` in order to have it join the cluster again.
7. Feel free to run further tests and to modify the cluster configuration (especially resources and constraints) in order to adapt the cluster to your individual needs. The rest is up to you now! ☺