By Falko Timme
Published: 2009-03-01 20:27

Xen Cluster Management With Ganeti On Debian Lenny

Version 1.0
Author: Falko Timme <ft [at] falkotimme [dot] com>
Last edited 02/26/2009

Ganeti is a cluster virtualization management system based on **Xen**. In this tutorial I will explain how to create one virtual Xen machine (called an *instance*) on a cluster of two physical nodes, and how to manage and failover this instance between the two physical nodes.

This document comes without warranty of any kind! I do not issue any guarantee that this will work for you!

1 Preliminary Note

In this tutorial I will use the physical nodes *node1.example.com* and *node2.example.com*:

- *node1.example.com*: IP address *192.168.0.100*; will be the master of the cluster.
- *node2.example.com*: IP address *192.168.0.101*; will be the primary node of the virtual machine (aka *instance*).

Both have a 500GB hard drive of which I use 20GB for the */* partition, 1GB for swap, and leave the rest unpartitioned so that it can be used by Ganeti (the minimum is 20GB!). Of course, you can change the partitioning to your liking, but remember about the minimum unused space.

The cluster I'm going to create will be named *cluster1.example.com*, and it will have the IP address *192.168.0.102*. The cluster IP *192.168.0.102* will always be bound to the cluster master, so even if you don't know which node is the master, you can use the cluster IP (or the hostname *cluster1.example.com*) to connect to the master using SSH.

The Xen virtual machine (called an *instance* in Ganeti speak) will be named *inst1.example.com* with the IP address *192.168.0.105*. *inst1.example.com* will be mirrored between the two physical nodes using **DRBD** - you can see this as a kind of network RAID1.

As you see, *node1.example.com* will be the cluster master, i.e. the machine from which you can control and manage the cluster, and *node2.example.com* will be the primary node of *inst1.example.com*, i.e. *inst1.example.com* will run on *node2.example.com* (with all changes on *inst1.example.com* mirrored back to *node1.example.com* with DRBD) until you fail it over to *node1.example.com* (if you want to take down *node2.example.com* for maintenance, for example). This is an active-passive configuration.

I think it's good practice to split up the roles between the two nodes, so that you don't lose the cluster master and the primary node at once should one node go down.

It is important that all hostnames mentioned here should be resolvable to all hosts, which means that they must either exist in DNS, or you must put all hostnames in all */etc/hosts* files on all hosts (which is what I will do here).

All cluster nodes must use the same network interface (e.g. *eth0*). If one node uses *eth0* and the other one *eth1*, then Ganeti won't work correctly anymore.

Ok, let's start...

2 Preparing The Physical Nodes

node1:

I want *node1* to have the static IP address *192.168.0.100*, therefore my */etc/network/interfaces* file looks as follows (please note that I replace *allow-hotplug eth0* with *auto eth0*; otherwise restarting the network doesn't work, and we'd have to reboot the whole system):

```
vi /etc/network/interfaces
```

```
# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
#allow-hotplug eth0
```

```
#iface eth0 inet dhcp
auto eth0
iface eth0 inet static
        address 192.168.0.100
        netmask 255.255.255.0
        network 192.168.0.0
        broadcast 192.168.0.255
        gateway 192.168.0.1
```

If you've modifed the file, restart your network:

```
/etc/init.d/networking restart
```

Then edit */etc/hosts*. Make it look like this:

```
vi /etc/hosts
```

```
127.0.0.1       localhost.localdomain   localhost
192.168.0.100   node1.example.com       node1
192.168.0.101   node2.example.com       node2
192.168.0.102   cluster1.example.com    cluster1
192.168.0.105   inst1.example.com       inst1

# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

```
ff02::3 ip6-allhosts
```

Next we must make sure that the commands

```
hostname
```

and

```
hostname -f
```

print out the full hostname (*node1.example.com*). If you get something different (e.g. just *node1*), do this:

```
echo node1.example.com > /etc/hostname

/etc/init.d/hostname.sh start
```

Afterwards, the *hostname* commands should show the full hostname.

Then update the system:

```
aptitude update
```

```
aptitude safe-upgrade
```

node2:

Now we do the same again on *node2.example.com* (please keep in mind that *node2* has a different IP!):

```
vi /etc/network/interfaces
```

```
# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
#allow-hotplug eth0
#iface eth0 inet dhcp
auto eth0
iface eth0 inet static
        address 192.168.0.101
        netmask 255.255.255.0
        network 192.168.0.0
        broadcast 192.168.0.255
        gateway 192.168.0.1
```

```
/etc/init.d/networking restart
```

```
vi /etc/hosts
```

```
127.0.0.1       localhost.localdomain   localhost
192.168.0.100   node1.example.com       node1
192.168.0.101   node2.example.com       node2
192.168.0.102   cluster1.example.com    cluster1
192.168.0.105   inst1.example.com       inst1
```

```
# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts
```

```
echo node2.example.com > /etc/hostname

/etc/init.d/hostname.sh start
```

```
aptitude update
```

```
aptitude safe-upgrade
```

3 Setting Up LVM On The Free HDD Space

node1/node2:

Let's find out about our hard drive:

```
fdisk -l
```

```
node1:~# fdisk -l

Disk /dev/sda: 500.1 GB, 500107862016 bytes
255 heads, 63 sectors/track, 60801 cylinders
```

```
Units = cylinders of 16065 * 512 = 8225280 bytes
Disk identifier: 0x00023cd1

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *           1          62      497983+  83  Linux
/dev/sda2              63        6141    48829567+  8e  Linux LVM
node1:~#
```

We will now create the partition */dev/sda3* (on both physical nodes) using the rest of the hard drive and prepare it for LVM:

```
fdisk /dev/sda
```

```
node1:~# fdisk /dev/sda

  The number of cylinders for this disk is set to 60801.
  There is nothing wrong with that, but this is larger than 1024,
  and could in certain setups cause problems with:
  1) software that runs at boot time (e.g., old versions of LILO)
  2) booting and partitioning software from other OSs
   (e.g., DOS FDISK, OS/2 FDISK)

Command (m for help): <-- n
Command action
   e   extended
   p   primary partition (1-4)
<-- p
Partition number (1-4): <-- 3
First cylinder (6142-60801, default 6142): <-- ENTER
Using default value 6142
Last cylinder or +size or +sizeM or +sizeK (6142-60801, default 60801): <-- ENTER
Using default value 60801
```

```
Command (m for help): <-- t
Partition number (1-4): <-- 3
Hex code (type L to list codes): <-- L

 0  Empty           1e  Hidden W95 FAT1 80  Old Minix       be  Solaris boot
 1  FAT12           24  NEC DOS         81  Minix / old Lin bf  Solaris
 2  XENIX root      39  Plan 9          82  Linux swap / So c1  DRDOS/sec (FAT-
 3  XENIX usr       3c  PartitionMagic  83  Linux           c4  DRDOS/sec (FAT-
 4  FAT16 <32M      40  Venix 80286     84  OS/2 hidden C:  c6  DRDOS/sec (FAT-
 5  Extended        41  PPC PReP Boot   85  Linux extended  c7  Syrinx
 6  FAT16           42  SFS             86  NTFS volume set da  Non-FS data
 7  HPFS/NTFS       4d  QNX4.x          87  NTFS volume set db  CP/M / CTOS / .
 8  AIX             4e  QNX4.x 2nd part 88  Linux plaintext de  Dell Utility
 9  AIX bootable    4f  QNX4.x 3rd part 8e  Linux LVM       df  BootIt
 a  OS/2 Boot Manag 50  OnTrack DM      93  Amoeba          e1  DOS access
 b  W95 FAT32       51  OnTrack DM6 Aux 94  Amoeba BBT      e3  DOS R/O
 c  W95 FAT32 (LBA) 52  CP/M            9f  BSD/OS          e4  SpeedStor
 e  W95 FAT16 (LBA) 53  OnTrack DM6 Aux a0  IBM Thinkpad hi eb  BeOS fs
 f  W95 Ext'd (LBA) 54  OnTrackDM6      a5  FreeBSD         ee  EFI GPT
10  OPUS            55  EZ-Drive        a6  OpenBSD         ef  EFI (FAT-12/16/
11  Hidden FAT12    56  Golden Bow      a7  NeXTSTEP        f0  Linux/PA-RISC b
12  Compaq diagnost 5c  Priam Edisk     a8  Darwin UFS      f1  SpeedStor
14  Hidden FAT16 <3 61  SpeedStor       a9  NetBSD          f4  SpeedStor
16  Hidden FAT16    63  GNU HURD or Sys ab  Darwin boot     f2  DOS secondary
17  Hidden HPFS/NTF 64  Novell Netware  b7  BSDI fs         fd  Linux raid auto
18  AST SmartSleep  65  Novell Netware  b8  BSDI swap       fe  LANstep
1b  Hidden W95 FAT3 70  DiskSecure Mult bb  Boot Wizard hid ff  BBT
1c  Hidden W95 FAT3 75  PC/IX
Hex code (type L to list codes): <-- 8e
Changed system type of partition 3 to 8e (Linux LVM)

Command (m for help): <-- w
The partition table has been altered!
```

```
Calling ioctl() to re-read partition table.

WARNING: Re-reading the partition table failed with error 16: Device or resource busy.
The kernel still uses the old table.
The new table will be used at the next reboot.
Syncing disks.
node1:~#
```

Now let's take a look at our hard drive again:

```
fdisk -l
```

```
node1:~# fdisk -l

Disk /dev/sda: 500.1 GB, 500107862016 bytes
255 heads, 63 sectors/track, 60801 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Disk identifier: 0x00023cd1

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *           1          62      497983+  83  Linux
/dev/sda2              63        6141    48829567+  8e  Linux LVM
/dev/sda3            6142       60801   439056450   8e  Linux LVM
node1:~#
```

Looks good. Now we must reboot both physical nodes so that the kernel can read in the new partition table:

```
reboot
```

After the reboot, we install LVM (probably it's already installed, but it's better to go sure):

```
aptitude install lvm2
```

After the reboot, we prepare */dev/sda3* for LVM on both nodes and add it to the volume group *xenvg*:

```
pvcreate /dev/sda3

vgcreate xenvg /dev/sda3
```

(Ganeti wants to use a volume group of its own, that's why we create *xenvg*; theoretically we could use an existing volume group with enough unallocated space, but the *gnt-cluster verify* command will complain about this.)

4 Installing Ganeti And Xen

node1/node2:

We can install Ganeti and Xen with one simple command:

```
aptitude install ganeti
```

You will see the following question:

MD arrays needed for the root file system: <-- all

Then we edit */etc/xen/xend-config.sxp* and modify the following settings:

```
vi /etc/xen/xend-config.sxp
```

```
[...]
```

```
(xend-relocation-server yes)
[...]
(xend-relocation-port 8002)
[...]
(xend-relocation-address '')
[...]
(network-script network-bridge)
[...]
#(network-script network-dummy)
[...]
(vif-script vif-bridge)
[...]
(dom0-min-mem 0)
[...]
```

Next open */boot/grub/menu.lst* and find the *# xenhopt=* and *# xenkopt=* lines and modify them as follows (don't remove the # at the beginning!):

```
vi /boot/grub/menu.lst
```

```
[...]
## Xen hypervisor options to use with the default Xen boot option
# xenhopt=dom0_mem=256M

## Xen Linux kernel options to use with the default Xen boot option
# xenkopt=console=tty0 nosmp
[...]
```

256M or 512M are a reasonable amount of memory for *dom0*.

(Please use `nosmp` only of your CPU has multiple cores. If you CPU has just one core, it is possible that it won't boot anymore with this setting. You can check how many cores you have with the following command:

```
cat /proc/cpuinfo
```

)

Afterwards, update the GRUB boot loader:

```
/sbin/update-grub
```

and reboot both physical nodes:

```
reboot
```

After the reboot, the nodes should run the Xen kernel:

```
uname -r
```

```
node1:~# uname -r
  2.6.26-1-xen-686
node1:~#
```

Afterwards do this:

```
cd /boot

ln -s vmlinuz-`uname -r` vmlinuz-2.6-xenU
```

```
ln -s initrd.img-`uname -r` initrd-2.6-xenU
```

(This is useful if you don't specify a kernel in the *gnt-instance add* command - the command will then use */boot/vmlinuz-2.6-xenU* and */boot/initrd-2.6-xenU* by default.)

5 Installing DRBD

node1/node2:

Next we install DRBD:

```
aptitude install drbd8-modules-`uname -r` drbd8-utils
```

Now we must enable the DRBD kernel module:

```
echo drbd minor_count=64 >> /etc/modules

modprobe drbd minor_count=64
```

It is recommended to configure LVM not to scan the DRBD devices. Therefore we open */etc/lvm/lvm.conf* and replace the *filter* line as follows:

```
vi /etc/lvm/lvm.conf
```

```
[...]
filter = [ "r|/dev/cdrom|", "r|/dev/drbd[0-9]+|" ]
[...]
```

6 Initializing The Cluster

node1:

Now we can initialize our cluster (this has to be done only once per cluster). Our clustername is *cluster1.example.com*, and I want *node1.example.com* to be the master, therefore we run the following command on *node1.example.com*:

```
gnt-cluster init -b eth0 -g xenvg --master-netdev eth0 cluster1.example.com
```

Ganeti assumes that the name of the volume group is *xenvg* by default, so you can also leave out the *-g xenvg* switch, but if your volume group has a different name, you must specify it with the *-g* switch.

Xen 3.2 and 3.3 don't use the bridge *xen-br0* anymore; instead *eth0* is used, therefore we must specify *-b eth0* and *--master-netdev eth0*.

7 Adding node2.example.com To The Cluster

node1:

Now that *node1* is the master, we run all commands for managing the cluster on *node1*. In order to add *node2.example.com* to the cluster, we run:

```
gnt-node add node2.example.com
```

This will look like this:

```
node1:~# gnt-node add node2.example.com
  -- WARNING --
  Performing this operation is going to replace the ssh daemon keypair
  on the target machine (node2.example.com) with the ones of the current one
  and grant full intra-cluster ssh root access to/from it

  The authenticity of host 'node2.example.com (192.168.0.101)' can't be established.
```

```
 RSA key fingerprint is 62:d3:d4:3f:d2:9c:3b:f2:5f:fe:c0:8a:c8:02:82:2a.
 Are you sure you want to continue connecting (yes/no)? <-- yes
root@node2.example.com's password: <-- node2's root password
node1:~#
```

Now let's check if our cluster really consists out of *node1* and *node2*:

```
gnt-node list
```

You should get something like this:

```
node1:~# gnt-node list
Node              DTotal  DFree MTotal MNode MFree Pinst Sinst
node1.example.com 428764 428764   3839   256  3535     0     0
node2.example.com 104452 104452   1023   256   747     0     0
node1:~#
```

8 Setting Up An Instance

node1:

Now let's create our first virtual machine (called an *instance* in Ganeti speak), *inst1.example.com*. I want to use DRBD for it (remote RAID1), I want *node2* to be the primary node, and I want the instance to have a 5 GB hard drive, 256 MB swap and 256 MB RAM. Again, we run the command on the cluster master, *node1.example.com*:

```
gnt-instance add -t drbd -n node2.example.com:node1.example.com -o debootstrap -s 5g --swap-size 256 -m 256 --kernel /boot/vmlinuz-`uname -r`
--ip 192.168.0.105 inst1.example.com
```

(I've specified *--kernel /boot/vmlinuz-`uname -r`*; if you don't specify a kernel, Ganeti will use */boot/vmlinuz-2.6-xenU* by default - see chapter 4.)

This can take some time. This is how the output looks:

```
node1:~# gnt-instance add -t drbd -n node2.example.com:node1.example.com -o debootstrap -s 5g --swap-size 256 -m 256 --kernel
 /boot/vmlinuz-`uname -r` --ip 192.168.0.105 inst1.example.com
* creating instance disks...
adding instance inst1.example.com to cluster config
 - INFO: Waiting for instance inst1.example.com to sync disks.
 - INFO: - device sda:  3.90% done, 971 estimated seconds remaining
 - INFO: - device sdb: 17.00% done, 42 estimated seconds remaining
 - INFO: - device sda:  9.00% done, 746 estimated seconds remaining
 - INFO: - device sdb: 100.00% done, 0 estimated seconds remaining
 - INFO: - device sda:  9.30% done, 727 estimated seconds remaining
 - INFO: - device sda: 22.10% done, 786 estimated seconds remaining
 - INFO: - device sda: 35.10% done, 224 estimated seconds remaining
 - INFO: - device sda: 48.00% done, 205 estimated seconds remaining
 - INFO: - device sda: 61.00% done, 183 estimated seconds remaining
 - INFO: - device sda: 73.90% done, 120 estimated seconds remaining
 - INFO: - device sda: 86.90% done, 36 estimated seconds remaining
 - INFO: - device sda: 94.80% done, 344 estimated seconds remaining
 - INFO: Instance inst1.example.com's disks are in sync.
creating os for instance inst1.example.com on node node2.example.com
* running the instance OS create scripts...
* starting instance...
node1:~#
```

Ganeti has created a complete virtual machine (using Debian Lenny) which you can now use.

9 Configuring The Instance

node1:

To get to *inst1.example.com*'s command line, run

```
gnt-instance console inst1.example.com
```

on *node1*.

You will notice that the console hangs, and you don't see a login prompt:

```
Checking file systems...fsck 1.41.3 (12-Oct-2008)
  done.
  Setting kernel variables (/etc/sysctl.conf)...done.
  Mounting local filesystems...done.
  Activating swapfile swap...done.
  Setting up networking....
  Configuring network interfaces...done.
  INIT: Entering runlevel: 2
  Starting enhanced syslogd: rsyslogd.
Starting periodic command scheduler: crond.
```

Shut down the instance...

```
gnt-instance shutdown inst1.example.com
```

... and start it with the *--extra "xencons=tty1 console=tty1"* parameter (do this everytime you start the instance):

```
gnt-instance startup --extra "xencons=tty1 console=tty1" inst1.example.com
```

Afterwards, connect to the console again...

```
gnt-instance console inst1.example.com
```

... and log in to *inst1.example.com*. The username is *root* along with no password. Therefore the first thing we do after the login is create a password for *root*:

inst1.example.com:

```
passwd
```

Next we must add a stanza for *eth0* to */etc/network/interfaces*. Right now, *inst1.example.com* has no network connectivity because only *lo* (the loopback interface) is up.

As I said in chapter 1, I want *inst1.example.com* to have the IP address *192.168.0.105*:

```
vi /etc/network/interfaces
```

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
        address 192.168.0.105
        netmask 255.255.255.0
        network 192.168.0.0
        broadcast 192.168.0.255
        gateway 192.168.0.1
```

Restart the network afterwards:

```
/etc/init.d/networking restart
```

Run

```
aptitude update
```

```
aptitude safe-upgrade
```

to update the instance, and then install OpenSSH and *vim-nox*:

```
aptitude install ssh openssh-server vim-nox udev
```

Before you connect to *inst1.example.com* using an SSH client such as **PuTTY**, open /etc/fstab...

```
vi /etc/fstab
```

... and add the following line (otherwise you will get the following error in your SSH client: *Server refused to allocate pty*):

```
[...]
none /dev/pts devpts gid=5,mode=620 0 0
```

Then run

```
mount -a
```

Now you can connect to *inst1.example.com* using an SSH client such as **PuTTY** on the IP address *192.168.0.105*.

To leave *inst1*'s console and get back to *node1*, type *CTRL+]* if you are at the console, or *CTRL+5* if you're using PuTTY (this is the same as if you were

using Xen's *xm* commands instead of Ganeti).

10 Further Ganeti Commands

To learn more about what you can do with Ganeti, take a look at the following man pages:

```
man gnt-instance
```

```
man gnt-cluster
```

```
man gnt-node
```

```
man gnt-os
```

```
man gnt-backup
```

```
man 7 ganeti
```

```
man 7 ganeti-os-interface
```

and also at the Ganeti administrator's guide that comes with the Ganeti package (in */docs/admin.html*). The **Ganeti installation tutorial** also has some hints.

The most interesting commands should be these:

Start an instance:

```
gnt-instance startup inst1.example.com
```

Stop an instance:

```
gnt-instance shutdown inst1.example.com
```

Go to an instance's console:

```
gnt-instance console inst1.example.com
```

Failover an instance to its secondary node (the instance will be stopped during this operation!):

```
gnt-instance failover inst1.example.com
```

Doing a live migration (i.e., the instance will keep running) to its secondary node:

```
gnt-instance migrate inst1.example.com
```

Delete an instance:

```
gnt-instance remove inst1.example.com
```

Get a list of instances:

```
gnt-instance list
```

```
node1:~# gnt-instance list
Instance          OS          Primary_node      Status  Memory
inst1.example.com debootstrap node2.example.com running    256
node1:~#
```

Get more details about instances:

```
gnt-instance info
```

```
node1:~# gnt-instance info
Instance name: inst1.example.com
State: configured to be up, actual state is up
Considered for memory checks in cluster verify: True
  Nodes:
    - primary: node2.example.com
    - secondaries: node1.example.com
  Operating system: debootstrap
  Kernel path: /boot/vmlinuz-2.6.26-1-xen-686
       initrd: (default: /boot/initrd-2.6-xenU)
  Hardware:
    - VCPUs: 1
    - memory: 256MiB
    - NICs: {MAC: aa:00:00:b5:00:8d, IP: 192.168.0.105, bridge: eth0}
  Block devices:
    - sda, type: drbd8, logical_id: (u'node2.example.com', u'node1.example.com', 11000)
      primary:   /dev/drbd0 (147:0) in sync, status ok
      secondary: /dev/drbd0 (147:0) in sync, status ok
      - type: lvm, logical_id: (u'xenvg', u'9c923acc-14b4-460d-946e-3b0d4d2e18e6.sda_data')
        primary:   /dev/xenvg/9c923acc-14b4-460d-946e-3b0d4d2e18e6.sda_data (253:2)
        secondary: /dev/xenvg/9c923acc-14b4-460d-946e-3b0d4d2e18e6.sda_data (253:2)
      - type: lvm, logical_id: (u'xenvg', u'4ffe2d67-584e-4581-9cd6-30da33c21b04.sda_meta')
        primary:   /dev/xenvg/4ffe2d67-584e-4581-9cd6-30da33c21b04.sda_meta (253:3)
```

```
        secondary: /dev/xenvg/4ffe2d67-584e-4581-9cd6-30da33c21b04.sda_meta (253:3)
    - sdb, type: drbd8, logical_id: (u'node2.example.com', u'node1.example.com', 11001)
      primary:   /dev/drbd1 (147:1) in sync, status ok
      secondary: /dev/drbd1 (147:1) in sync, status ok
      - type: lvm, logical_id: (u'xenvg', u'4caff02e-3864-47b3-ba58-b71854a7b7c0.sdb_data')
        primary:   /dev/xenvg/4caff02e-3864-47b3-ba58-b71854a7b7c0.sdb_data (253:4)
        secondary: /dev/xenvg/4caff02e-3864-47b3-ba58-b71854a7b7c0.sdb_data (253:4)
      - type: lvm, logical_id: (u'xenvg', u'51fb132b-083e-42e2-aefa-31fd485a8aab.sdb_meta')
        primary:   /dev/xenvg/51fb132b-083e-42e2-aefa-31fd485a8aab.sdb_meta (253:5)
        secondary: /dev/xenvg/51fb132b-083e-42e2-aefa-31fd485a8aab.sdb_meta (253:5)
node1:~#
```

Get info about a cluster:

```
gnt-cluster info
```

```
node1:~# gnt-cluster info
  Cluster name: cluster1.example.com
  Master node: node1.example.com
  Architecture (this node): 32bit (i686)
  Cluster hypervisor: xen-3.0
node1:~#
```

Check if everything is alright with the cluster:

```
gnt-cluster verify
```

```
node1:~# gnt-cluster verify
  * Verifying global settings
  * Gathering data (2 nodes)
  * Verifying node node1.example.com
```



```
  * Verifying node node2.example.com
  * Verifying instance inst1.example.com
  * Verifying orphan volumes
  * Verifying remaining instances
  * Verifying N+1 Memory redundancy
  * Other Notes
  * Hooks Results
node1:~#
```

Find out who's the cluster master:

```
gnt-cluster getmaster
```

```
node1:~# gnt-cluster getmaster
  node1.example.com
node1:~#
```

Failover the master if the master has gone down (fails over the master to the node on which this command is run):

```
gnt-cluster masterfailover
```

Find out about instance volumes on the cluster nodes:

```
gnt-node volumes
```

```
node1:~# gnt-node volumes
Node              PhysDev   VG    Name                                          Size Instance
node1.example.com /dev/sda2 vg0   root                                         28608 -
node1.example.com /dev/sda2 vg0   swap_1                                         952 -
node1.example.com /dev/sda3 xenvg 4caff02e-3864-47b3-ba58-b71854a7b7c0.sdb_data   256 inst1.example.com
```

```
node1.example.com /dev/sda3 xenvg 4ffe2d67-584e-4581-9cd6-30da33c21b04.sda_meta   128 inst1.example.com
node1.example.com /dev/sda3 xenvg 51fb132b-083e-42e2-aefa-31fd485a8aab.sdb_meta   128 inst1.example.com
node1.example.com /dev/sda3 xenvg 9c923acc-14b4-460d-946e-3b0d4d2e18e6.sda_data  5120 inst1.example.com
node2.example.com /dev/hda2 vg0   root                                          28608 -
node2.example.com /dev/hda2 vg0   swap_1                                          952 -
node2.example.com /dev/hda3 xenvg 4caff02e-3864-47b3-ba58-b71854a7b7c0.sdb_data   256 inst1.example.com
node2.example.com /dev/hda3 xenvg 4ffe2d67-584e-4581-9cd6-30da33c21b04.sda_meta   128 inst1.example.com
node2.example.com /dev/hda3 xenvg 51fb132b-083e-42e2-aefa-31fd485a8aab.sdb_meta   128 inst1.example.com
node2.example.com /dev/hda3 xenvg 9c923acc-14b4-460d-946e-3b0d4d2e18e6.sda_data  5120 inst1.example.com
node1:~#
```

Removing a node from a cluster:

```
gnt-node remove node2.example.com
```

Find out about the operating systems supported by the cluster (currently only *debootstrap*):

```
gnt-os list
```

```
node1:~# gnt-os list
  Name
  debootstrap
node1:~#
```

11 A Failover Example

Now let's assume you want to take down *node2.example.com* due to maintenance and you therefore want to fail over *inst1.example.com* to *node1* (please note that *inst1.example.com* will be shut down during the failover, but will be switched on again instantly thereafter) .

First, let's find out about our instances:

node1:

```
gnt-instance list
```

As you see, *node2* is the primary node:

```
node1:~# gnt-instance list
Instance          OS          Primary_node      Status  Memory
inst1.example.com debootstrap node2.example.com running    256
node1:~#
```

To failover *inst1.example.com* to *node1*, we run the following command (again on *node1*):

```
gnt-instance failover inst1.example.com
```

```
node1:~# gnt-instance failover inst1.example.com
  Failover will happen to image inst1.example.com. This requires a
  shutdown of the instance. Continue?
  y/[n]/?: <-- y
 * checking disk consistency between source and target
  * shutting down instance on source node
  * deactivating the instance's disks on source node
  * activating the instance's disks on target node
  * starting the instance on the target node
node1:~#
```

Afterwards, we run

```
gnt-instance list
```

again. *node1* should now be the primary node:

```
node1:~# gnt-instance list
Instance          OS          Primary_node      Status  Memory
inst1.example.com debootstrap node1.example.com running    256
node1:~#
```

As *inst1.example.com* has started again immediately after the failover, we need to fix the console problem again (see chapter 9):

```
gnt-instance shutdown inst1.example.com
```

```
gnt-instance startup --extra "xencons=tty1 console=tty1" inst1.example.com
```

Now you can take down *node2*:

node2:

```
shutdown -h now
```

After *node2* has gone down, you can try to connect to *inst1.example.com* - it should still be running.

Now after the maintenance on *node2* is finished and we have booted it again, we'd like to make it the primary node again.

Therefore we try a failover on *node1* again:

node1:

```
gnt-instance failover inst1.example.com
```

This time we get this:

```
node1:~# gnt-instance failover inst1.example.com
  Failover will happen to image inst1.example.com. This requires a
  shutdown of the instance. Continue?
  y/[n]/?: <-- y
 * checking disk consistency between source and target
  Node node2.example.com: Disk degraded, not found or node down
  Failure: command execution error:
  Disk sda is degraded on target node, aborting failover.
node1:~#
```

The failover doesn't work because *inst1.example.com*'s hard drive on *node2* is degraded (i.e., not in sync).

To fix this, we can replace *inst1.example.com*'s disks on *node2* by mirroring the disks from the current primary node, *node1*, to *node2*:

node1:

```
gnt-instance replace-disks -s inst1.example.com
```

During this process (which can take some time) *inst1.example.com* can stay up.

```
node1:~# gnt-instance replace-disks -s inst1.example.com
STEP 1/6 check device existence
 - INFO: checking volume groups
 - INFO: checking sda on node2.example.com
 - INFO: checking sda on node1.example.com
 - INFO: checking sdb on node2.example.com
 - INFO: checking sdb on node1.example.com
STEP 2/6 check peer consistency
 - INFO: checking sda consistency on node1.example.com
 - INFO: checking sdb consistency on node1.example.com
```

```
STEP 3/6 allocate new storage
 - INFO: creating new local storage on node2.example.com for sda
 - INFO: creating new local storage on node2.example.com for sdb
STEP 4/6 change drbd configuration
 - INFO: detaching sda drbd from local storage
 - INFO: renaming the old LVs on the target node
 - INFO: renaming the new LVs on the target node
 - INFO: adding new mirror component on node2.example.com
 - INFO: detaching sdb drbd from local storage
 - INFO: renaming the old LVs on the target node
 - INFO: renaming the new LVs on the target node
 - INFO: adding new mirror component on node2.example.com
STEP 5/6 sync devices
 - INFO: Waiting for instance inst1.example.com to sync disks.
 - INFO: - device sda:  1.80% done, 560 estimated seconds remaining
 - INFO: - device sdb: 12.40% done, 35 estimated seconds remaining
 - INFO: - device sda:  5.80% done, 832 estimated seconds remaining
 - INFO: - device sdb: 89.30% done, 3 estimated seconds remaining
 - INFO: - device sda:  6.40% done, 664 estimated seconds remaining
 - INFO: - device sdb: 98.50% done, 0 estimated seconds remaining
 - INFO: - device sda:  6.50% done, 767 estimated seconds remaining
 - INFO: - device sdb: 100.00% done, 0 estimated seconds remaining
 - INFO: - device sda:  6.50% done, 818 estimated seconds remaining
 - INFO: - device sda: 19.30% done, 387 estimated seconds remaining
 - INFO: - device sda: 32.00% done, 281 estimated seconds remaining
 - INFO: - device sda: 44.70% done, 242 estimated seconds remaining
 - INFO: - device sda: 57.30% done, 195 estimated seconds remaining
 - INFO: - device sda: 70.00% done, 143 estimated seconds remaining
 - INFO: - device sda: 82.70% done, 74 estimated seconds remaining
 - INFO: - device sda: 95.40% done, 20 estimated seconds remaining
 - INFO: - device sda: 99.80% done, 3 estimated seconds remaining
 - INFO: Instance inst1.example.com's disks are in sync.
STEP 6/6 removing old storage
```

```
 - INFO: remove logical volumes for sda
 - INFO: remove logical volumes for sdb
node1:~#
```

Afterwards, we can failover *inst1.example.com* to *node2*:

```
gnt-instance failover inst1.example.com
```

node2 should now be the primary again:

```
gnt-instance list
```

```
node1:~# gnt-instance list
Instance          OS          Primary_node      Status  Memory
inst1.example.com debootstrap node2.example.com running    256
node1:~#
```

(Now do this again:

```
gnt-instance shutdown inst1.example.com
```

```
gnt-instance startup --extra "xencons=tty1 console=tty1" inst1.example.com
```

)

12 A Live Migration Example

One of the great Ganeti features is that you can do live migrations of instances, i.e., you can move them from one node to the other without taking them down (live migration works only if you're using DRBD 0.8, it doesn't work with DRBD 0.7).

To migrate *inst1.example.com* from *node2* to *node1*, we run:

node1:

```
gnt-instance migrate inst1.example.com
```

```
node1:~# gnt-instance migrate inst1.example.com
  Instance inst1.example.com will be migrated. Note that migration is
  **experimental** in this version. This might impact the instance if
  anything goes wrong. Continue?
  y/[n]/?: <-- y
 * checking disk consistency between source and target
  * identifying disks
  * switching node node1.example.com to secondary mode
  * changing into standalone mode
  * changing disks into dual-master mode
  * wait until resync is done
  * migrating instance to node1.example.com
  * switching node node2.example.com to secondary mode
  * wait until resync is done
  * changing into standalone mode
  * changing disks into single-master mode
  * wait until resync is done
  * done
node1:~#
```

The command

```
gnt-instance list
```

should now show that *inst1.example.com* is now running on *node1*:

```
node1:~# gnt-instance list
Instance          OS          Primary_node      Status  Memory
inst1.example.com debootstrap node1.example.com running    256
node1:~#
```

Let's migrate it back to *node2*:

```
gnt-instance migrate inst1.example.com
```

```
node1:~# gnt-instance migrate inst1.example.com
  Instance inst1.example.com will be migrated. Note that migration is
  **experimental** in this version. This might impact the instance if
  anything goes wrong. Continue?
  y/[n]/?: <-- y
 * checking disk consistency between source and target
  * identifying disks
  * switching node node2.example.com to secondary mode
  * changing into standalone mode
  * changing disks into dual-master mode
  * wait until resync is done
  * migrating instance to node2.example.com
  * switching node node1.example.com to secondary mode
  * wait until resync is done
  * changing into standalone mode
  * changing disks into single-master mode
  * wait until resync is done
  * done
node1:~#
```

```
gnt-instance list
```

```
node1:~# gnt-instance list
Instance          OS          Primary_node      Status  Memory
inst1.example.com debootstrap node2.example.com running    256
node1:~#
```

13 Creating A Backup Of An Instance

To create a backup of *inst1.example.com* on *node1*, we run (the instance will be shut down during this operation!):

node1:

```
gnt-backup export -n node1.example.com inst1.example.com
```

The backup will be stored in the */var/lib/ganeti/export/inst1.example.com/* directory:

```
ls -l /var/lib/ganeti/export/inst1.example.com/
```

```
node1:~# ls -l /var/lib/ganeti/export/inst1.example.com/
total 108788
-rw-r--r-- 1 root root 111279899 2009-02-26 17:30 9c923acc-14b4-460d-946e-3b0d4d2e18e6.sda_data.snap
-rw------- 1 root root       391 2009-02-26 17:30 config.ini
node1:~#
```

To export the backup to another cluster node, e.g. *node3*, we run

```
gnt-backup import -n node3.example.com -t drbd --src-node=node1.example.com --src-dir=/var/lib/ganeti/export/inst1.example.com/
inst1.example.com
```

14 Masterfailover

Now let's assume our cluster master, *node1*, has gone down for whatever reason. Therefore we need a new master. To make *node2* the new cluster master, we run the following command on *node2*:

node2:

```
gnt-cluster masterfailover
```

```
node2:~# gnt-cluster masterfailover
  caller_connect: could not connect to remote host node1.example.com, reason [Failure instance: Traceback (failure with no
frames): <class 'twisted.internet.error.ConnectError'>: An error occurred while connecting: 113: No route to host.
  ]
  could disable the master role on the old master node1.example.com, please disable manually
  caller_connect: could not connect to remote host node1.example.com, reason [Failure instance: Traceback (failure with no
frames): <class 'twisted.internet.error.ConnectError'>: An error occurred while connecting: 113: No route to host.
  ]
  caller_connect: could not connect to remote host node1.example.com, reason [Failure instance: Traceback (failure with no
frames): <class 'twisted.internet.error.ConnectError'>: An error occurred while connecting: 113: No route to host.
  ]
node2:~#
```

Now run

```
gnt-cluster getmaster
```

to verify that *node2* is the new master:

```
node2:~# gnt-cluster getmaster
  node2.example.com
node2:~#
```

Now when *node1* comes up again, we have a split-brain situation - *node1* thinks it is the master...

node1:

```
gnt-cluster getmaster
```

```
node1:~# gnt-cluster getmaster
  node1.example.com
node1:~#
```

... while in fact *node2* is the master.

To fix this, we edit */var/lib/ganeti/ssconf_master_node* on node1:

node1:

```
chmod 600 /var/lib/ganeti/ssconf_master_node

vi /var/lib/ganeti/ssconf_master_node
```

```
node2.example.com
```

```
chmod 400 /var/lib/ganeti/ssconf_master_node
```

Afterwards,...

```
gnt-cluster getmaster
```

... shows the right master:

```
node1:~# gnt-cluster getmaster
  node2.example.com
node1:~#
```

To make *node1* the master again, just run

```
gnt-cluster masterfailover
```

on *node1* - if both *node1* and *node2* are running during this operation, both will know that *node1* is the new master afterwards.

15 Links

- Ganeti: **http://code.google.com/p/ganeti**
- Xen: **http://xen.xensource.com**
- DRBD: **http://www.drbd.org**
- LVM: **http://sourceware.org/lvm2**
- Debian: **http://www.debian.org**