By GHALEB
Published: 2008-06-10 19:09

How To Set Up A Loadbalanced High-Availability Apache Cluster Based On Ubuntu 8.04 LTS

Pleasenote that my main reference and source is Falko's article here just modified for Ubuntu 8.04 LTS. Also some parts were taken from Falko's article "The Perfect Server - Ubuntu 8.04 LTS" ***here***.

Version 1.0
Author: Mohamed Ghaleb <Mohamed_Ghaleb [at] msn [dot] com> (English and German only please)
Last edited 06/03/2008

This tutorial shows how to set up a two-node Apache web servercluster that provides high-availability. In front of the Apache clusterwe create a load balancer that splits up incoming requests between thetwo Apache nodes. Because we do not want the load balancer to becomeanother "Single Point Of Failure", we must provide high-availabilityfor the load balancer, too. Therefore our load balancer will in factconsist out of two load balancer nodes that monitor each other using `heartbeat`, and if one loadbalancer fails, the other takes over silently.

The advantage of using a load balancer compared to using ***roundrobin DNS***is that it takes care of the load on the web server nodes and tries todirect requests to the node with less load, and it also takes care ofconnections/sessions. Many web applications (e.g. forum software,shopping carts, etc.) make use of sessions, and if you are in a sessionon Apache node 1, you would lose that session if suddenly node 2 servedyour requests. In addition to that, if one of the Apache nodes goesdown, the load balancer realizes that and directs all incoming requeststo the remaining node which would not be possible with round robin DNS.

For this setup, we need four nodes (two Apache nodes and twoload balancer nodes) and ***five***IP addresses: one for each node and one virtual IP address that will beshared by the load balancer nodes and used for incoming HTTP requests.

I will use the following setup here:

- Apache node 1: `webserver1.tm.local`(`webserver1`)- IP address: `192.168.0.103`;Apache document root: `/var/www`
- Apache node 2: `webserver2.tm.local`(`webserver2`)- IP address: `192.168.0.104`;Apache document root: `/var/www`

- Load Balancer node 1: *loadb1.tm.local*(*loadb1*) - IP address: *192.168.0.101*
- Load Balancer node 2: *loadb2.tm.local*(*loadb2*) - IP address: *192.168.0.102*
- Virtual IP Address: *192.168.0.105*(used for incoming requests)

Have a look at the drawing on ***http://www.linuxvirtualserver.org/docs/ha/ultramonkey.html***to understand how this setup looks like.

In this tutorial I will use ***Ubuntu 8.04 LTS***for all four nodes, just install basic Ubuntu 8.04 LTS on all four nodes.

I want to say first that this is not the only way of settingup sucha system. There are many ways of achieving this goal but this is theway I take. I do not issue any guarantee that this will work for you!

I also recommend you to have a DNS server in place.

Step 1 to 6 should be done on all four servers.

1 Enable The root Account

Run

```
sudo passwd root
```

and give root a password. Afterwards we become root by running

```
su
```

2 Install The SSH Server

If you did not install the OpenSSH server during the system installation, you can do it now:

```
apt-get install ssh openssh-server
```

From now on you can use an SSH client such as ***PuTTY*** and connect from your workstation to your Ubuntu 8.04 LTS server and follow the remaining steps from this tutorial.

3 Install vim-full

I'll use `vi` as my text editor in this tutorial. The default `vi` program has some strange behavior on Ubuntu and Debian; to fix this, we install `vim-full`:

```
apt-get install vim-full
```

(You don't have to do this if you use a different text editor such as joe or nano.)

4 Configure The Network

Because the Ubuntu installer has configured our system to get itsnetwork settings via DHCP, we have to change that now because a servershould have a static IP address. Edit ***/etc/network/interfaces*** and adjust it to your needs (in this example setup I will use the IP address `192.168.0.101`):

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

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
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
```

Please make sure your network configurationare set correctly, feel free to change that based on your networkconfiguration.

Then 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
192.168.0.101   loadb1.tm.local         loadb1

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

Now run

```
echo loadb1.tm.local > /etc/hostname
```

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

Afterwards, run

```
hostname

hostname -f
```

Both should show `loadb1.tm.local` now.

If you have a DNS server in place (recommended) make sure the 4 servers configured to use it, if you don't have a DNS ***click here***

```
vi /etc/resolv.conf
```

```
search tm.local
nameserver 192.168.0.100
```

5 Edit /etc/apt/sources.list And Update Your Linux Installation

Edit `/etc/apt/sources.list`. Comment out or remove the installation CD from the file and make sure that the `universe` and `multiverse` repositories are enabled. It should look like this:

```
vi /etc/apt/sources.list
```

```
#
# deb cdrom:[Ubuntu-Server 8.04 _Hardy Heron_ - Release i386 (20080423.2)]/ hardy main restricted
#deb cdrom:[Ubuntu-Server 8.04 _Hardy Heron_ - Release i386 (20080423.2)]/ hardy main restricted
# See http://help.ubuntu.com/community/UpgradeNotes for how to upgrade to
# newer versions of the distribution.
deb http://de.archive.ubuntu.com/ubuntu/ hardy main restricted
deb-src http://de.archive.ubuntu.com/ubuntu/ hardy main restricted
## Major bug fix updates produced after the final release of the
## distribution.
deb http://de.archive.ubuntu.com/ubuntu/ hardy-updates main restricted
deb-src http://de.archive.ubuntu.com/ubuntu/ hardy-updates main restricted
## N.B. software from this repository is ENTIRELY UNSUPPORTED by the Ubuntu
## team, and may not be under a free licence. Please satisfy yourself as to
## your rights to use the software. Also, please note that software in
## universe WILL NOT receive any review or updates from the Ubuntu security
## team.
deb http://de.archive.ubuntu.com/ubuntu/ hardy universe
deb-src http://de.archive.ubuntu.com/ubuntu/ hardy universe
deb http://de.archive.ubuntu.com/ubuntu/ hardy-updates universe
deb-src http://de.archive.ubuntu.com/ubuntu/ hardy-updates universe
## N.B. software from this repository is ENTIRELY UNSUPPORTED by the Ubuntu
## team, and may not be under a free licence. Please satisfy yourself as to
## your rights to use the software. Also, please note that software in
## multiverse WILL NOT receive any review or updates from the Ubuntu
## security team.
deb http://de.archive.ubuntu.com/ubuntu/ hardy multiverse
deb-src http://de.archive.ubuntu.com/ubuntu/ hardy multiverse
deb http://de.archive.ubuntu.com/ubuntu/ hardy-updates multiverse
deb-src http://de.archive.ubuntu.com/ubuntu/ hardy-updates multiverse
## Uncomment the following two lines to add software from the 'backports'
## repository.
## N.B. software from this repository may not have been tested as
## extensively as that contained in the main release, although it includes
```

```
## newer versions of some applications which may provide useful features.
## Also, please note that software in backports WILL NOT receive any review
## or updates from the Ubuntu security team.
# deb http://de.archive.ubuntu.com/ubuntu/ hardy-backports main restricted universe multiverse
# deb-src http://de.archive.ubuntu.com/ubuntu/ hardy-backports main restricted universe multiverse
## Uncomment the following two lines to add software from Canonical's
## 'partner' repository. This software is not part of Ubuntu, but is
## offered by Canonical and the respective vendors as a service to Ubuntu
## users.
# deb http://archive.canonical.com/ubuntu hardy partner
# deb-src http://archive.canonical.com/ubuntu hardy partner
deb http://security.ubuntu.com/ubuntu hardy-security main restricted
deb-src http://security.ubuntu.com/ubuntu hardy-security main restricted
deb http://security.ubuntu.com/ubuntu hardy-security universe
deb-src http://security.ubuntu.com/ubuntu hardy-security universe
deb http://security.ubuntu.com/ubuntu hardy-security multiverse
deb-src http://security.ubuntu.com/ubuntu hardy-security multiverse
```

Then run

```
apt-get update
```

to update the apt package database and

```
apt-get upgrade
```

to install the latest updates (if there are any).

6 Disable AppArmor

AppArmor is a security extension (similar to SELinux) that shouldprovide extended security, which usually causes more problems than advantages. Therefore Idisable it.

We can disable it like this:

```
/etc/init.d/apparmor stop

update-rc.d -f apparmor remove
```

7 Install Apache (Only on the Webservers)

webserver1/webserver2:

```
apt-get install apache2
```

8 Enable IPVS On The Load Balancers

First we must enable IPVS on our load balancers. IPVS (IPVirtual Server) implements transport-layer load balancing inside theLinux kernel, so called Layer-4 switching.

loadb1/loadb2:

```
echo ip_vs_dh >>
/etc/modules

echo ip_vs_ftp >> /etc/modules

echo ip_vs >> /etc/modules

echo ip_vs_lblc >> /etc/modules
```

```
echo ip_vs_lblcr >> /etc/modules

echo ip_vs_lc >> /etc/modules

echo ip_vs_nq >> /etc/modules

echo ip_vs_rr >> /etc/modules

echo ip_vs_sed >> /etc/modules

echo ip_vs_sh >> /etc/modules

echo ip_vs_wlc >> /etc/modules

echo ip_vs_wrr >> /etc/modules
```

Then we do this:

loadb1/loadb2:

```
modprobe ip_vs_dh

modprobe ip_vs_ftp

modprobe ip_vs

modprobe ip_vs_lblc

modprobe ip_vs_lblcr

modprobe ip_vs_lc
```

```
modprobe ip_vs_nq

modprobe ip_vs_rr

modprobe ip_vs_sed

modprobe ip_vs_sh

modprobe ip_vs_wlc

modprobe ip_vs_wrr
```

9 Install Ultra Monkey (packages) On The Load Balancers

UltraMonkeyis a project to create load balanced and highly available services on alocal area network using Open Source components on the Linux operatingsystem; the Ultra Monkey package provides *heartbeat*(used by the two load balancers to monitor each other and check if theother node is still alive) and *ldirectord*,the actual load balancer.

Inthe Original article Falko uses Debian and thus there are directlyDebian repositories from Ultra Monkey, however as here we are usingUbuntu we will have to install ipvsadm ldirectord heartbeat.

loadb1/loadb2:

```
apt-get install ipvsadm ldirectord heartbeat
```

If you see this warning:

```
Â¦ libsensors3 not functional                                Â¦
Â¦                                                           Â¦
Â¦ It appears that your kernel is not compiled with sensors support. As a   Â¦
```

```
Â¦ result, libsensors3 will not be functional on your system.  Â¦
Â¦                                                             Â¦
Â¦ If you want to enable it, have a look at "I2C Hardware Sensors Chip  Â¦
Â¦ support" in your kernel configuration.
```

you can ignore it, I didn't see it on Ubuntu, but as it was in the original article I though to include it just in case.

10 Enable Packet Forwarding On The Load Balancers

The load balancers must be able to route traffic to theApachenodes. Therefore we must enable packet forwarding on the loadbalancers. Add the following lines to */etc/sysctl.conf*:

loadb1/loadb2:

```
vi /etc/sysctl.conf
```

```
# Enables packet forwarding
net.ipv4.ip_forward = 1
```

Then do this:

loadb1/loadb2:

```
sysctl -p
```

11 Configure heartbeat And ldirectord

Now we have to create three configuration files for *heartbeat*. They must beidentical on *loadb1* and*loadb2*!

loadb1/loadb2:

```
vi /etc/ha.d/ha.cf
```

```
logfacility local0
bcast eth0 # Linux
mcast eth0 225.0.0.1 694 1 0
auto_failback off
node loadb1.tm.local
node loadb2.tm.local
respawn hacluster /usr/lib/heartbeat/ipfail
apiauth ipfail gid=haclient uid=hacluster
```

Important: As nodenames we must use theoutput of

```
uname -n
```

on *loadb1*and *loadb2*.

loadb1/loadb2:

```
vi /etc/ha.d/haresources
```

```
loadb1.tm.local \
 ldirectord::ldirectord.cf \
 LVSSyncDaemonSwap::master \
 IPaddr2::192.168.0.105/24/eth0/192.168.0.255
```

Please note that the last line above has my virtual IP which is: 192.168.0.105, my netmask is 255.255.255.0 and as its class C my IP should be followed by /24 then at the end my broadcast IP 192.168.0.255, please make sure you use the correct IP configuration.

The first word in the first line above is the output of

```
uname -n
```

This file should be the same on both nodes, no matter if you start to create the file on *loadb1*or *loadb2*! After *IPaddr2* we put our virtual IPaddress *192.168.0.105*.

loadb1/loadb2:

```
vi /etc/ha.d/authkeys
```

```
auth 3
3 md5 somerandomstring
```

*somerandomstring*is a password which the two *heartbeat*daemons on *loadb1* and *loadb2*use to authenticate against each other. Use your own string here. Youhave the choice between three authentication mechanisms. I use *md5* as I believe it is the most secureone.

*/etc/ha.d/authkeys*should be readable by root only, therefore we do this:

loadb1/loadb2:

```
chmod 600 /etc/ha.d/authkeys
```

ldirectord isthe actual load balancer. We are going to configure our two loadbalancers (*loadb1.tm.local*and *loadb2.tm.local*)in an ***active/passive*** setup, which means we have one active load balancer, and the other oneis a hot-standby and becomes active if the active one fails. To make itwork, we must

create the *ldirectord*configuration file */etc/ha.d/ldirectord.cf*which again must be identical on *loadb1*and *loadb2*.

loadb1/loadb2:

```
vi /etc/ha.d/ldirectord.cf
```

```
checktimeout=10
checkinterval=2
autoreload=no
logfile="local0"
quiescent=yes

virtual=192.168.0.105:80
 real=192.168.0.103:80 gate
 real=192.168.0.104:80 gate
 fallback=127.0.0.1:80 gate
 service=http
 request="ldirector.html"
 receive="Test Page"
 scheduler=rr
 protocol=tcp
 checktype=negotiate
```

In the *virtual*=line we put our virtual IP address (*192.168.0.105*in this example), and in the *real*=lines we list the IP addresses of our Apache nodes (*192.168.0.103* and *192.168.0.104* in this example).In the *request=* line welist the name of a file on *webserver1*and *webserver2* that *ldirectord* will requestrepeatedly to see if *webserver1*and *webserver2* arestill alive. That file (that we are going to create later on) mustcontain the string listed in the *receive*=line.

Afterwards we create the system startup links for *heartbeat* and remove those of *ldirectord* because *ldirectord* will be started bythe *heartbeat* daemon:

loadb1/loadb2:

```
update-rc.d heartbeat start 75 2 3 4
5 . stop 05 0 1 6 .

update-rc.d -f ldirectord remove
```

Finally we start *heartbeat*(and with it *ldirectord*):

loadb1/loadb2:

```
/etc/init.d/ldirectord stop

/etc/init.d/heartbeat start
```

12 Test The Load Balancers

Let's check if both load balancers work as expected:

loadb1/loadb2:

```
ip addr sh eth0
```

The active load balancer should list the virtual IP address(*192.168.0.105*):

```
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 00:0c:29:4e:67:1a brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.101/24 brd 192.168.0.255 scope global eth0
    inet 192.168.0.105/24 brd 192.168.0.255 scope global secondary eth0
```

The hot-standby should show this:

```
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 00:0c:29:34:d7:7e brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.102/24 brd 192.168.0.255 scope global eth0
```

loadb1/loadb2:

```
ldirectord ldirectord.cf status
```

Output on the active load balancer:

```
ldirectord for /etc/ha.d/ldirectord.cf is running with pid: 5321
```

Output on the hot-standby:

```
ldirectord is stopped for /etc/ha.d/ldirectord.cf
```

loadb1/loadb2:

```
ipvsadm -L -n
```

Output on the active load balancer:

```
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port           Forward Weight ActiveConn InActConn
TCP  192.168.0.105:80 rr
  -> 192.168.0.103:80             Route   1      0          0
  -> 192.168.0.104:80             Route   0      0          0
```

Output on the hot-standby:

```
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port           Forward Weight ActiveConn InActConn
```

loadb1/loadb2:

```
/etc/ha.d/resource.d/LVSSyncDaemonSwap
master status
```

Output on the active load balancer:

```
master running
(ipvs_syncmaster pid: 5470)
```

Output on the hot-standby:

```
master stopped
```

If your tests went fine, you can now go on and configure thetwo Apache nodes.

13 Configure The Two Apache Nodes

Finally we must configure our Apache cluster nodes *webserver1.tm.local* and *webserver2.tm.local* to acceptrequests on the virtual IP address *192.168.0.105*.

webserver1/webserver2:

```
apt-get install iproute
```

Add the following to */etc/sysctl.conf*:

webserver1/webserver2:

```
vi /etc/sysctl.conf
```

```
# Enable configuration of arp_ignore option
net.ipv4.conf.all.arp_ignore = 1

# When an arp request is received on eth0, only respond if that address is
# configured on eth0. In particular, do not respond if the address is
# configured on lo
net.ipv4.conf.eth0.arp_ignore = 1

# Ditto for eth1, add for all ARPing interfaces
#net.ipv4.conf.eth1.arp_ignore = 1



# Enable configuration of arp_announce option
```

```
net.ipv4.conf.all.arp_announce = 2

# When making an ARP request sent through eth0 Always use an address that
# is configured on eth0 as the source address of the ARP request. If this
# is not set, and packets are being sent out eth0 for an address that is on
# lo, and an arp request is required, then the address on lo will be used.
# As the source IP address of arp requests is entered into the ARP cache on
# the destination, it has the effect of announcing this address. This is
# not desirable in this case as adresses on lo on the real-servers should
# be announced only by the linux-director.
net.ipv4.conf.eth0.arp_announce = 2

# Ditto for eth1, add for all ARPing interfaces
#net.ipv4.conf.eth1.arp_announce = 2
```

Then run this:

webserver1/webserver2:

```
sysctl -p
```

Add this section for the virtual IP address to */etc/network/interfaces*:

webserver1/webserver2:

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

```
auto lo:0
iface lo:0 inet static
```

```
address 192.168.0.105
netmask 255.255.255.255
pre-up sysctl -p > /dev/null
```

Then run this:

Please Note after the following step you will probably get this error: SIOCSIFFLAGS: Cannot assign requested address

That is a normal ***bug*** and you can ignore it.

webserver1/webserver2:

```
ifup lo:0
```

If you change the IP at a later stage its recommended to do ***ifup lo:0*** then ***ifdown lo:0*** then again ***ifup lo:0***

Finally we must create the file `ldirector.html`.This file is requested by the two load balancer nodes repeatedly sothat they can see if the two Apache nodes are still running. I assumethat the document root of the main apache web site on `webserver1` and `webserver2` is `/var/www`, therefore we createthe file `/var/www/ldirector.html`:

webserver1/webserver2:

```
vi /var/www/ldirector.html
```

```
Test Page
```

14 Further Testing

You can now access the web site that is hosted by the twoApache nodes by typing *http://192.168.0.105*in your browser.

Now stop the Apache on either *webserver1*or *webserver2*. Youshould then still see the web site on *http://192.168.0.105*because the load balancer directs requests to the working Apache node.Of course, if you stop both Apaches, then your request will fail.

Now let's assume that *loadb1*is our active load balancer, and *loadb2*is the hot-standby. Now stop *heartbeat*on *loadb1*:

loadb1:

```
/etc/init.d/heartbeat stop
```

Wait a few seconds, and then try *http://192.168.0.105*again in your browser. You should still see your web site because *loadb2* has taken the active rolenow.

Now start heartbeat again on loadb1:

loadb1:

```
/etc/init.d/heartbeat start
```

loadb2 shouldstill have the active role. Do the tests from chapter 5 again on *loadb1* and *loadb2*, and you should see theinverse results as before.

If you have also passed these tests, then your loadbalancedApache cluster is working as expected. Have fun!

15 Further Reading

This tutorial shows how to loadbalance two Apache nodes. Itdoes notshow how to keep the files in the Apache document root in sync or howto create a storage solution like an NFS server that both Apache nodescan use, nor does it provide a solution how to manage your MySQLdatabase(s). You can find solutions for these issues here:

- ***MirrorYour Web Site With rsync***

- ***Setting Up A Highly Available NFS Server***
- ***How To Set Up A Load-Balanced MySQL Cluster***
- ***How To Set Up Database Replication In MySQL***

16 Links

- heartbeat / The High-Availability Linux Project: ***http://linux-ha.org***
- The Linux Virtual Server Project: ***http://www.linuxvirtualserver.org***
- Ultra Monkey: ***http://www.ultramonkey.org***

17 References and Sources

- Falko's Article "***The Perfect Server - Ubuntu Hardy Heron (Ubuntu 8.04 LTS Server)***"
- Falko's Article "***How to Set up A Loadbalanced High-Availability Apache Cluster***"
- BIND on Ubuntu "***Howto: Setup a DNS server with bind***"