English | Deutsch Log in or Sign up

HowtoForge LINUX TUTORIALS

Tutorials Tags Forums Linux Commands Subscribe ISPConfig News

Tutorial search

Tutorials › How To Set Up A Loadbalanced High-Availability Apache Cluster

This is a *"copy & paste"* HowTo! The easiest way to follow this tutorial is to use a command line client/SSH client (like PuTTY for Windows) and simply copy and paste the commands (except where you have to provide own information like IP addresses, hostnames, passwords,...). This helps to avoid typos.

On this page

How To Set Up A Loadbalanced High-Availability Apache Cluster

Version 1.0
Author: Falko Timme

This tutorial shows how to set up a two-node Apache web server cluster that provides high-availability. In front of the Apache cluster we create a load balancer that splits up incoming requests between the two Apache nodes. Because we do not want the load balancer to become another "Single Point Of Failure", we must provide high-availability for the load balancer, too. Therefore our load balancer will in fact consist out of two load balancer nodes that monitor each other using *heartbeat*, and if one load balancer fails, the other takes over silently.

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

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

I will use the following setup here:

- Apache node 1: *webserver1.example.com* (*webserver1*) - IP address: *192.168.0.101*; Apache document root: */var/www*
- Apache node 2: *webserver2.example.com* (*webserver2*) - IP address: *192.168.0.102*; Apache document root: */var/www*
- Load Balancer node 1: *loadb1.example.com* (*loadb1*) - IP address: *192.168.0.103*
- Load Balancer node 2: *loadb2.example.com* (*loadb2*) - IP address: *192.168.0.104*
- 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 **Debian Sarge** for all four nodes. I assume that you have installed a basic Debian installation on all four nodes, and that you have installed Apache on *webserver1* and *webserver2*, with */var/www* being the document root of the main web site.

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

1 Enable IPVS On The Load Balancers

First we must enable IPVS on our load balancers. IPVS (IP Virtual Server) implements transport-layer load balancing inside the Linux 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
```

If you get errors, then most probably your kernel wasn't compiled with IPVS support, and you need to compile a new kernel with IPVS support (or install a kernel image with IPVS support) now.

2 Install Ultra Monkey On The Load Balancers

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

To install Ultra Monkey, we must edit `/etc/apt/sources.list` now and add these two lines (don't remove the other repositories):

loadb1/loadb2:

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

```
deb http://www.ultramonkey.org/download/3/ sarge main
deb-src http://www.ultramonkey.org/download/3 sarge main
```

Afterwards we do this:

loadb1/loadb2:

```
apt-get update
```

and install Ultra Monkey:

loadb1/loadb2:

```
apt-get install ultramonkey
```

If you see this warning:

```
¦ libsensors3 not functional                                          ¦
¦                                                                     ¦
¦ It appears that your kernel is not compiled with sensors support. As a
¦ result, libsensors3 will not be functional on your system.         ¦
¦                                                                     ¦
¦ If you want to enable it, have a look at "I2C Hardware Sensors Chip ¦
¦ support" in your kernel configuration.                              ¦
```

you can ignore it.

During the Ultra Monkey installation you will be asked a few question. Answer as follows:

`Do you want to automatically load IPVS rules on boot?`
<-- **No**

`Select a daemon method.`
<-- **none**

Next >>

view as pdf | print

Share this page: Tweet Follow @howtoforgecom 27.8K followers Recommend 17 G+1 11

Sub pages

Suggested articles

10 Comment(s)

Add comment

Name *

Email *

p

I'm not a robot

reCAPTCHA
Privacy - Terms

Submit comment

Comments

From: Anonymous Reply

thank you falko, it is the best guide.
Regards
noahlau

From: Reply

I must prove it in Ubuntu 6.06

....

thks.

From: John Wards Reply

Just a note to say that this tutorial works in 8.10 i386 server.
I believe it won't work in x64.

From: Jimmy Reply

This was one of the best cluster guides I have seen. Just wanted to say thank you. We are using it on onelight.

From: Anonymous Reply

How hard would it be to set up ISPConfig in a similar fashion?

From: Reply

Tested and approved with Debian GNU/Linux 2.6.26 and backports.
I'm wondering why there are loaded both of the scheduling "lblc" and "lblcr" ? If we have only two nodes in our cluster then cannot be talk for a "server set" and replication scheduling will be senseless.
Correct me if i'm wrong.
With respect: Imagandi

From: Anonymous Reply

Hello Everyone.. sorry if this is not the best place to ask, but I was just wondering..
Will I need a Layer4 Switch to have all this settings working?
If I do configure my nodes as above without it (switch) will I be able to still have a load-balanced Apache without it?
Thanks so much, and appreciate the effort to make this tutorial. 😀
See ya.

From: Anonymous Reply

Thanks for this article............

From: Anonymous Reply

Layer 3*

And no the switches "think" the ip is on both of the load balancers. It doesnt matter where it gets to since the load balancers will make sure the request gets to a node.

read : http://www.linuxvirtualserver.org/docs/ha/ultramonkey.html

From: Mahbubul Hasan Laskar

Reply

Hello Everyone.
I have got three tomcat servers and a front end apache httpd server which is also the load balancer. I want similar configuration but my apache httpd servers run on Windows OS. Ultramonkey does not work on windows and I have not found anything else. Is it possible to acheive multiple load balancers running on Windows OS?

Thanks

Tutorials **How To Set Up A Loadbalanced High-Availability Apache Cluster**

Tutorial Info

Author:	falko
Tags:	apache, debian, high-availability