安装部署 nginx 负载均衡

作者：junhunbaoke
原文链接：https://ld246.com/article/1601708510413
来源网站：链滴

使用两台虚拟机部署nginx：

安装依赖包：

<pre class="brush:csharp;gutter:true;"># yum install gcc gcc-c++ openssl-devel pcre-devel zlib-devel
</pre>

上传源码包到root下：

<pre class="brush:csharp;gutter:true;">nginx-1.12.2.tar.gz
</pre>

解压并进入nginx：

<pre class="brush:csharp;gutter:true;"># tar zxf nginx-1.12.2.tar.gz
cd nginx-1.12.2</pre>

编译并安装：

<pre class="brush:csharp;gutter:true;"># ./configure && make && make install</pre>

启动nginx：

<pre class="brush:csharp;gutter:true;"># /usr/local/nginx/sbin/nginx
</pre>

查看80端口是否启动成功：

<pre class="brush:csharp;gutter:true;"># netstat -lptnu | grep 80
</pre>

启动成功查看网页是否显示nginx：

安装keepalived：

<pre class="brush:csharp;gutter:true;"># yum install keepalived</pre>

修改其配置文件：

<pre class="brush:csharp;gutter:true;"># vi /etc/keepalived/keepalived.conf

! Configuration File for keepalived

global_defs {
   router_id LVS_DEVEL
}

vrrp_script check_nginx {
  script "pidof nginx"
  interval 2
  weight 20
}

vrrp_instance check_nginx {
    state BACKUP
    interface ens37
    virtual_router_id 98
    priority 99
    virtual_ipaddress {
        192.168.197.100/24
    }
    track_script {
        check_nginx
    }
}</pre>

启动keepalived:

<pre class="brush:csharp;gutter:true;"># systemctl start keepalived</pre>

查看是否配置成功:

<pre class="brush:csharp;gutter:true;"># ip a</pre>

```
ens37: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
 link/ether 00:0c:29:0a:11:1d brd ff:ff:ff:ff:ff:ff
 inet 192.168.197.129/24 brd 192.168.197.255 scope global noprefixroute dynamic ens37
    valid_lft 1743sec preferred_lft 1743sec
 inet 192.168.197.100/24 scope global secondary ens37
    valid_lft forever preferred_lft forever
 inet6 fe80::f80d:72ed:2cab:961c/64 scope link noprefixroute
    valid_lft forever preferred_lft forever
```

```
ens37: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
 link/ether 00:0c:29:fe:5e:03 brd ff:ff:ff:ff:ff:ff
 inet 192.168.197.128/24 brd 192.168.197.255 scope global noprefixroute dynamic ens37
    valid_lft 1181sec preferred_lft 1181sec
 inet6 fe80::4ff8:701f:547e:be4d/64 scope link noprefixroute
    valid_lft forever preferred_lft forever
```

关闭第一台nginx:

<pre class="brush:csharp;gutter:true;"># netstat -lptnu |grep 80
tcp   0   0 0.0.0.0:80   0.0.0.0:*   LISTEN   3631/nginx: master
tcp   0   0 0.0.0.0:22   0.0.0.0:*   LISTEN   1080/sshd
tcp6   0   0 :::22   :::*   LISTEN   1080/sshd
kill 3631

```
# netstat -lptnu |grep 80
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      1080/sshd
tcp6       0      0 :::22                   :::*                    LISTEN      1080/sshd</pre>
```

进行测试：查看 nginx VIP 是否跳转

```
ens37: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
 link/ether 00:0c:29:0a:11:1d brd ff:ff:ff:ff:ff:ff
 inet 192.168.197.129/24 brd 192.168.197.255 scope global noprefixroute dynamic ens37
    valid_lft 1598sec preferred_lft 1598sec
 inet6 fe80::f80d:72ed:2cab:961c/64 scope link noprefixroute
    valid_lft forever preferred_lft forever
```

```
ens37: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
 link/ether 00:0c:29:fe:5e:03 brd ff:ff:ff:ff:ff:ff
 inet 192.168.197.128/24 brd 192.168.197.255 scope global noprefixroute dynamic ens37
    valid_lft 1712sec preferred_lft 1712sec
 inet 192.168.197.100/24 scope global secondary ens37
    valid_lft forever preferred_lft forever
 inet6 fe80::4ff8:701f:547e:be4d/64 scope link noprefixroute
    valid_lft forever preferred_lft forever
```

nginx 常见的负载均衡有哪几种：

1. 轮询（默认）

轮询是默认的方式，轮询的方法是通过按照时间顺序将请求往不同的后端服务器发送，来缓解服器的压力。如果后台服务器上某一台宕机了，它可以自动剔除。缺点：可靠性低和负载分配不均衡。用于图片服务器和静态页面服务器集群。

2. weight

指定轮询几率，weight和访问比率成正比，用于后端服务器性能不均的情况。

```
<pre class="brush:csharp;gutter:true;">例如：  <br/>upstream bakend {
    server 192.168.159.10 weight=10;
    server 192.168.159.11 weight=10;
} </pre>
```

3. ip_hash

根据每个请求的ip的hash结果分配，因此每个固定ip能访问到同一个后端服务器，可以解决sessin问题。

```
<pre class="brush:csharp;gutter:true;">例如：  <br/>upstream resinserver{
    ip_hash;
    server 192.168.159.10:8080;
    server 192.168.159.11:8080;
} </pre>
```

4. fair（第三方）

按照后端服务器的相应时间来分配请求，时间短的优先分配。

```
<pre class="brush:csharp;gutter:true;">例如：<br/>upstream resinserver{
    server server1;
    server server2;
    fair;
} </pre>
```

5. url_hash（第三方）

按照访问url的hash结果来分配请求，每个固定的url访问同一个后端服务器。如果后端服务器是存时效率高。

<pre class="brush:csharp;gutter:true;">例如：在upstream中加入hash语句，server语句中不能入weight等其他的参数，hash_method是使用的hash算法：
upstream resinserver{
server squid1:3128;
server squid2:3128;
hash $request_uri;
hash_method crc32;
}

tips:
upstream resinserver{ #定义负载均衡设备的IP及设备状态
ip_hash;
server 127.0.0.1:8000 down;
server 127.0.0.1:8080 weight=2;
server 127.0.0.1:6801;
server 127.0.0.1:6802 backup;
}
</pre>

Nginx不仅仅是一款优秀的负载均衡器/反向代理软件，它同时也是功能强大的Web应用服务器，可做七层的转发 URL和目录的转发都可以做：

<pre class="brush:php;gutter:true;">nginx工作在网络的第7层，所以它可以针对http应用本身来分流策略，比如针对域名、目录结构等
nginx对网络的依赖较小，理论上只要ping得通，网页访问正常，nginx就能连得通
nginx安装和配置比较简单，测试起来也很方便
nginx可以检测到服务器内部的故障，比如根据服务器处理网页返回的状态码、超时等等，并且会把回错误的请求重新提交到另一个节点。</pre>