

第1章 Haproxy介绍

1.官方地址

```
1 | https://www.haproxy.com/
```

2.Haproxy和Nginx区别

- 1 haproxy只做反向代理负载均衡功能
- 2 Nginx除了能做代理转发，也可以做web服务器

3.Haproxy应用场景

- 1 4层负载
- 2 7层负载

第2章 Haproxy实战

1.Haproxy安装部署

注意：yum安装的haproxy有两个版本，名字不一样，默认haproxy为1.5版本，而最新的1.8版本名称为haproxy18

```
1 | #查找yum仓库里的haproxy版本
2 | [root@lb-5 ~]# yum search haproxy|grep ^haproxy
3 | haproxy18.x86_64 : HAProxy reverse proxy for high availability environments
4 | haproxy.x86_64 : TCP/HTTP proxy and load balancer for high availability
5 |
6 | #安装haproxy
7 | [root@lb-5 ~]# yum install haproxy -y
8 |
9 | #查看haproxy版本
10 | [root@lb-5 ~]# haproxy -v
11 | HA-Proxy version 1.5.18 2016/05/10
12 | Copyright 2000-2016 Willy Tarreau <willy@haproxy.org>
13 |
14 | #查看haproxy配置文件有哪些
15 | [root@lb-5 ~]# rpm -qc haproxy
16 | /etc/haproxy/haproxy.cfg
17 | /etc/logrotate.d/haproxy
18 | /etc/sysconfig/haproxy
```

安装1.8版本命令

```
1 | rpm -ivh http://www.nosuchhost.net/~cheese/fedora/packages/epel-
2 | 7/x86_64/cheese-release-7-1.noarch.rpm
3 | yum install haproxy
4 | haproxy -v
```

2.Haproxy配置文件解释

官方文档：需翻墙

1 | <http://www.haproxy.org/download/1.4/doc/configuration.txt>

配置说明:

- 1 | HAProxy配置中分成五部分内容，当然这些组件不是必选的，可以根据需要选择部分作为配置。
- 2 | **global**: 参数是进程级的，通常和操作系统相关。这些参数一般只设置一次，如果配置无误，就不需要再次配置进行修改
- 3 | **defaults**: 配置默认参数的，这些参数可以被利用配置到**frontend**, **backend**, **listen**组件
- 4 | **frontend**: 接收请求的前端虚拟节点，**Frontend**可以根据规则直接指定具体使用后端的**backend**(可动态选择)。
- 5 | **backend**: 后端服务集群的配置，是真实的服务器，一个**Backend**对应一个或者多个实体服务器。
- 6 | **listen**: **Frontend**和**Backend**的组合体。

全剧配置说明:

```
1 global
2     log             127.0.0.1 local2           #log
3     chroot          /var/lib/haproxy          #锁定运行目录
4     pidfile         /var/run/haproxy.pid      #pid文件路径
5     maxconn         4000                      #每个进程的最大并发数
6     user            haproxy                  #haproxy运行用户
7     group           haproxy                  #haproxy运行用户组
8     daemon          #以后台进程模式
运行
9     stats socket /var/lib/haproxy/stats      #socket文件路径
```

代理配置-defaults说明:

```
1 defaults
2     mode            http                    #默认运行的模式，有http
和tcp两种
3     log             global                 #日志路径，这里表示以global
定义的为准
4     option          httplog                #丰富日志格式，包含更多信息
5     option          dontlognull            #日志不记录空连接
6     option http-server-close              #在服务端启用HTTP连
接关闭
7     option forwardfor except 127.0.0.0/8  #启用X-Forwarded-For标头，记录
用户真实IP
8     option          redispatch             #在连接失败的情况下启用或禁用会话
重新分发
9     retries         3                     #设置连接失败后在服
务器上执行的重试次数
10    timeout http-request 10s                #设置等待完整的HTTP请求
所允许的最长时间
11    timeout queue    1m                    #设置在队列中等待空闲连
接插槽的最长时间
12    timeout connect  10s                   #设置等待连接服务器成功
的最长时间
13    timeout client   1m                    #在客户端上设置最长不活
动时间
```

14	timeout server	1m	#在服务器端设置最大不活动时间
15	timeout http-keep-alive	10s	#设置等待新HTTP请求出现的最长时间
16	timeout check	10s	#设置其他检查超时，但仅在已经建立连接之后
17	maxconn	3000	#将每个进程的最大并发连接数

代理配置-frontend说明:

```

1 frontend http_80_in
2     bind 0.0.0.0:80    #监听端口
3     mode http          #http的7层模式
4     log global          #应用全局的日志配置
5     option httplog      #启用http的log
6     option httpclose    #每次请求完毕后主动关闭http通道，HA-Proxy不支持keep-alive模式
7     option forwardfor   #获得客户端IP记录到forwardfor字段

```

代理配置-backend说明:

```

1 backend mms_server
2     mode http          #工作在http的7层模式
3     balance roundrobin #负载均衡的方式，roundrobin为默认平均轮询
4     option             #配置选项可以配置以下的参数用于后端健康检查
5         httpchk
6         mysql-check
7         pgsql-check
8         ssl-hello-chk
9     server             #后端服务器的配置可以配置以下参数
10        check          #健康检查
11        inter N        #健康检查间隔时间
12        fall N         #后端服务器失败检查次数
13        rise N         #后端服务器恢复检查次数
14        weight         #权重，0表示不参负载均衡

```

代理配置-listens说明:

```

1 listen web
2     bind 10.0.0.5:80
3     mode http
4     balance roundrobin
5     server web-7 10.0.0.7:80 check inter 3000 fall 3 rise 5
6     server web-8 10.0.0.8:80 check inter 3000 fall 3 rise 5

```

3.Haproxy代理实验

3.1 实验环境说明

1	1b-5	haproxy代理服务器
2	web-7	后端web服务器
3	web-8	后端web服务器

3.2 准备后端nginx测试环境

```
1 echo $(hostname) > /usr/share/nginx/html/index.html
2 systemctl restart nginx
3 curl 127.0.0.1
```

3.3 创建haproxy配置文件

```
1 cat > /etc/haproxy/haproxy.cfg << 'EOF'
2 global
3     log          127.0.0.1 local2
4     chroot       /var/lib/haproxy
5     pidfile      /var/run/haproxy.pid
6     maxconn      4000
7     user         haproxy
8     group        haproxy
9     daemon
10    stats socket /var/lib/haproxy/stats
11
12 defaults
13     mode          http
14     log           global
15     option        httplog
16     option        dontlognull
17     option http-server-close
18     option forwardfor except 127.0.0.0/8
19     option        redispatch
20     retries       3
21     timeout http-request 10s
22     timeout queue 1m
23     timeout connect 10s
24     timeout client 1m
25     timeout server 1m
26     timeout http-keep-alive 10s
27     timeout check 10s
28     maxconn       3000
29
30 frontend web_in_80
31     bind 10.0.0.5:80
32     mode http
33     use_backend web_server_80
34
35 backend web_server_80
36     mode http
37     option forwardfor
38     server web7 10.0.0.7:80 check inter 3000 fall 3 rise 5
39     server web8 10.0.0.8:80 check inter 3000 fall 3 rise 5
40
41 #listen可以把frontend和backend整合在一起，写起来更简洁
42 listen web_server_8080
43     bind 10.0.0.5:8080
44     mode http
45     balance roundrobin
46     server web7 10.0.0.7:8080 check inter 3000 fall 3 rise 5
47     server web8 10.0.0.8:8080 check inter 3000 fall 3 rise 5
48 EOF
```

3.4 启动Haproxy

```
1 | systemctl start haproxy
```

3.5 访问并测试

```
1 | curl 10.0.0.5
```

第3章 Haproxy调度算法

1.调度算法介绍

```
1 | Haproxy拥有很多中调度算法
```

2.基于权重的负载

```
1 |
```

3.基于轮询的负载

```
1 |
```

4.使用socat工具动态下线/上线后端节点

4.1 socat 工具介绍

- 1 | socat是一个功能强大的网络调试工具，支持众多协议，比如socket,IP,TCP等。
- 2 | 我们可以使用socat对haproxy的sokcat文件进行操作，使用haproxy支持的命令动态的上/下线节点，优势是不需要修改配置文件。

4.2 socat命令使用

```
1 | #!.安装工具
2 | yum -y install socat
3 |
4 | #2.查看命令帮助
5 | socat -h
6 |
7 | #3.查看haproxy的socket命令帮助
8 | echo "help" | socat stdio /var/lib/haproxy/haproxy.sock
9 |
10 | #4.下线节点
11 | echo disable server web/web-7| socat stdio /var/lib/haproxy/haproxy.sock
12 |
13 | #5.上线节点
14 | echo enable server web/web-8| socat stdio /var/lib/haproxy/haproxy.sock
```

4.3 jenkins代码上线滚动更新脚本

```
1  #!/bin/bash
2
3  PATH_CODE=/var/lib/jenkins/workspace/my-freestyle-job
4  PATH_WEB=/usr/share/nginx
5  if [ "$NODE" == "ALL" ];then
6      IP="10.0.0.7 10.0.0.8"
7  elif [ "$NODE" == "10.0.0.7" ];then
8      IP="10.0.0.7"
9  else
10     IP="10.0.0.8"
11 fi
12
13 #打包代码
14 code_tar(){
15     cd ${PATH_CODE}
16     tar zcf /opt/web-${git_version}.tar.gz ./
17 }
18
19 #负载均衡器下线节点
20 node_down(){
21     ssh root@10.0.0.5 "echo disable server web/${NODE}| socat stdio
22 /var/lib/haproxy/haproxy.sock"
23     echo "${NODE} 从 10.0.0.5 移除成功"
24 }
25
26 #拷贝打包好的代码发送到web服务器代码目录
27 code_scp(){
28     ssh ${NODE} "mkdir ${PATH_WEB}/web-${git_version} -p"
29     scp /opt/web-${git_version}.tar.gz
30     ${NODE}:${PATH_WEB}/web-${git_version}
31 }
32
33 #web服务器解压代码
34 code_xf(){
35     ssh ${NODE} "cd ${PATH_WEB}/web-${git_version} && tar xf web-
36 ${git_version}.tar.gz && rm -rf web-${git_version}.tar.gz"
37 }#创建代码软链接
38 code_ln(){
39     ssh ${NODE} "cd ${PATH_WEB} && rm -rf html && ln -s
40 web-${git_version} html"
41 }
42
43 #测试访问是否正常
44 web_test(){
45     curl -s -I -m 10 -o /dev/null -w %{http_code}
46 http://${NODE}/index.html
47 }
48
49 #负载均衡上线节点
50 node_up(){
51     ssh root@10.0.0.5 "echo enable server web/${NODE}| socat
52 stdio /var/lib/haproxy/haproxy.sock"
53
54     echo "${NODE} 从 10.0.0.5 添加成功"
```

```

50 }
51
52
53 #选择发布还是回滚
54 if [ "${deploy_env}" == "deploy" ];then
55     code_tar
56     for NODE in ${IP};do
57         node_down
58         code_scp
59         code_xf
60         code_ln
61         web_test
62         node_up
63     done
64 elif [ "${deploy_env}" == "rollback" ];then
65     for NODE in ${IP};do
66         node_down
67         code_ln
68         web_test
69         node_up
70     done
71 fi

```

第4章 ACL匹配

1.ACL配置说明

文字说明：

- 1 ACL(Access Control List)访问控制列表
- 2 是一种可以根据匹配的条件进行不同的跳转实现方法，比如可以根据浏览器类型，IP地址，URL，文件后缀名等条件进行匹配和转发。
- 3
- 4 在haproxy中，需要先定义ACL，然后再引用对应的ACL

配置语法：

- 1 `acl` 规则名称 匹配条件 匹配模式 执行操作 操作的对象

匹配模式说明：

- 1 `-i` 不区分大小写
- 2 `-m` 使用指定的pattern
- 3 `-n` 不做DNS解析

2.ACL匹配语法

2.1 hdr 请求报文首部匹配条件

匹配说明

```
1 | hdr_dom    匹配请求host名称 如 www.0ldboyedu.com
2 | hdr_beg    匹配请求host开头 如 www. bbs. blog
3 | hdr_end    匹配请求host开头结尾 如 .com .org
4 | hdr_sub
```

举例:

```
1 |
```

2.2 path 请求URL路径匹配条件

匹配说明:

```
1 | path_beg    匹配URL开头
2 | path_end    匹配URL结尾
3 | path_reg    匹配URL里的字符串
4 | path_dir    匹配路径
```

举例:

```
1 |
```

2.3 基于IP和端口匹配

匹配说明:

```
1 | dst          目标IP
2 | dst_port     目标PORT
3 | src          源IP
4 | src_port     源PORT
```

举例:

```
1 |
```

3.匹配规则实战

3.1 基于浏览器类型匹配

```
1 | #前端配置
2 | frontend web_in_80
3 |     bind 10.0.0.5:80
4 |     mode http
5 |     use_backend web_server_80
6 |
7 | #acl设置
8 | acl acl_user_agent_mobile hdr_sub(User-Agent) -i iphone Android
9 | acl acl_user_agent_pc hdr_sub(User-Agent) -i chrome
10 | acl acl_user_agent_linux hdr_sub(User-Agent) -i curl wget
11 |
12 | #acl调用
13 | use_backend mobile_hosts if acl_user_agent_mobile
```



```

14     use_backend linux_hosts if acl_user_agent_linux
15     default_backend pc_hosts if acl_user_agent_pc
16
17 #后端配置
18 backend mobile_hosts
19     mode http
20     server web1 10.0.0.7:80 check inter 3000 fall 3 rise 5
21
22 backend linux_hosts
23     mode http
24     server web1 10.0.0.8:80 check inter 3000 fall 3 rise 5
25
26 backend pc_hosts
27     mode http
28     server web1 10.0.0.7:80 check inter 3000 fall 3 rise 5
29     server web1 10.0.0.8:80 check inter 3000 fall 3 rise 5

```

3.2 基于文件后缀名匹配

```

1  #前端配置
2  frontend web_in_80
3      bind 10.0.0.5:80
4      mode http
5      use_backend web_server_80
6
7      #acl设置
8      acl acl_img path_end -i .jpg .png
9
10     #acl调用
11     use_backend static_hosts if acl_img
12     default_backend web_hosts
13
14 #后端配置
15 backend static_hosts
16     mode http
17     server web1 10.0.0.7:80 check inter 3000 fall 3 rise 5
18
19 backend web_hosts
20     mode http
21     server web1 10.0.0.7:80 check inter 3000 fall 3 rise 5
22     server web1 10.0.0.8:80 check inter 3000 fall 3 rise 5

```

3.3 基于域名匹配

```

1  #前端配置
2  frontend web_in_80
3      bind 10.0.0.5:80
4      mode http
5      use_backend web_server_80
6
7      #acl设置
8      acl acl_img path_end -i .jpg .png
9
10     #acl调用
11     use_backend static_hosts if acl_img
12     default_backend web_hosts

```

```

13
14 #后端配置
15 backend static_hosts
16     mode http
17     server web1 10.0.0.7:80 check inter 3000 fall 3 rise 5
18
19 backend web_hosts
20     mode http
21     server web1 10.0.0.7:80 check inter 3000 fall 3 rise 5
22     server web1 10.0.0.8:80 check inter 3000 fall 3 rise 5

```

3.4 基于IP匹配

```

1 #前端配置
2 frontend web_in_80
3     bind 10.0.0.5:80
4     mode http
5     use_backend web_server_80
6
7     #acl设置
8     acl acl_ip_172 src 172.16.1.0/24
9
10    #acl调用
11    use_backend 172_hosts if acl_ip_172
12
13 #后端配置
14 backend 172_hosts
15     mode http
16     server web1 172.16.1.7:80 check inter 3000 fall 3 rise 5

```

第4章 Haproxy高级功能

1.web服务状态监控

配置启动监控状态页面：

```

1 listen stats
2     bind :8080 #监听端口
3     stats enable #启用状态页
4     stats hide-version #隐藏版本号
5     stats uri /haproxy-status #自定义状态页路径
6     stats realm HAProxy\ Statistics\ Page #自定义提示信息
7     stats auth oldboy:123456 #定义普通用户密码
8     stats auth admin:123456 #定义admin用户密码
9     stats admin if TRUE #启用管理功能

```

访问并查看状态页面：

← → ↺ 不安全 10.0.0.5:8080/haproxy-status

HAProxy

Statistics Report for pid 9522

> General process information

pid = 9522 (process #1, nproc = 1)

uptime = 0d 0h0m36s

system limits: memmax = unlimited, ulimit-n = 8034

maxsock = 8034, maxconn = 4000, maxpipes = 0

current conn = 2, current pipes = 0/0, conn rate = 1/sec

Running tasks: 1/0, idle = 100 %

active UP

active UP, going down

active DOWN, going up

active or backup DOWN

active or backup DOWN for maintenance (MAINT)

active or backup SOFT STOPPED for maintenance

backup UP

backup UP, going down

backup DOWN, going up

not checked

Note: "NOLB?"DRAIN" = UP with load-balancing disabled.

Display option:

Scope :

Hide DOWN servers

Refresh now

CSV export

External resources:

Primary site

Updates v1.5

Online manual

web_in_80

	Queue			Session rate			Sessions				Bytes		Denied		Errors		Warnings		Status	LastChk	Wght	Server							
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn				Resp	Retr	Redis	Act	Bck	Chk	Dwn	Downtime
Frontend				0	0	-	0	0		3 000	0		0	0	0	0	0	0	0	0	0	0	OPEN						

web_server_80

	Queue			Session rate			Sessions				Bytes		Denied		Errors		Warnings		Status	LastChk	Wght	Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn				Resp	Retr	Redis	Act	Bck	Chk	Dwn	Downtime	Thrtle
<input type="checkbox"/> web7	0	0	-	0	0		0	0		-	0	0	?	0	0	0	0	0	0	0	0	0	36s UP	L4OK in 0ms	1	Y	-	0	0	-
<input type="checkbox"/> web8	0	0	-	0	0		0	0		-	0	0	?	0	0	0	0	0	0	0	0	0	36s UP	L4OK in 0ms	1	Y	-	0	0	-
Backend	0	0		0	0		0	0		300	0	0	?	0	0	0	0	0	0	0	0	0	36s UP		2	2	0		0	0s

Choose the action to perform on the checked servers:

▼

Apply

stats

	Queue			Session rate			Sessions				Bytes		Denied		Errors		Warnings		Status	LastChk	Wght	Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn				Resp	Retr	Redis	Act	Bck	Chk	Dwn	Downtime	Thrtle
Frontend				1	2	-	2	2		3 000	5		1 270	691	0	0	0	0	0	0	0	0	OPEN							
Backend	0	0		0	2		0	1		300	2	0	0s	1 270	691	0	0	2	0	0	0	0	36s UP		0	0	0		0	

状态字段解释：

1 |

TCP/IP四层模型	TCP/IP包含协议
应用层	HTTP SSH TELNET
传输层	TCP UDP
网络层	IPv4 IPv6 ARP ICMP
网络接入层	网络接口及硬件层

四层负载与七层负载区别：

- 1 四层负载:
- 2 所谓四层负载均衡指的是OSI七层模型中的传输层
- 3 Nginx已经能支持TCP/IP的控制, 所以只需要对客户端的请求进行TCP/IP协议的包转发就可以实现负载均衡
- 4 它的好处是性能非常快、只需要底层进行应用处理, 而不需要进行一些复杂的逻辑
- 5
- 6 七层负载均衡:
- 7 七层负载是指应用层, 它可以完成很多应用方面的协议请求
- 8 比如我们说的http应用的负载均衡, 它可以实现http信息的改写、头信息的改写、安全应用规则控制、URL匹配规则控制、以及转发,rewrite等等的规则
- 9 所以在应用层的服务里面, 我们可以做的内容就更多, 那么Nginx则是一个典型的七层负载均衡

haproxy四层负载配置:

```
1 listen redis-server
2     bind 10.0.0.5:6379
3     mode tcp
4     balance leastconn
5     server redis1 10.0.0.51:6379 check
6     server redis2 10.0.0.52:6379 check backup
```

haproxy四层负载访问测试: 前提是后端服务器安装好了redis并配置允许远程访问

```
1 redis-cli 10.0.0.5
2 >keys *
```