

若我若鱼

刘鸿鹤的个人网站，工作、生活，点点滴滴...

博客

Internet Tools

Archives

AllTags

April 1, 2016 Author: 若我若鱼

haproxy 的安装和配置

安装

```
1. tar zxvf haproxy-1.6.4.tar.gz
2. cd haproxy-1.6.4
3. make TARGET=linux26 PREFIX=/usr/local/haproxy
4. make install PREFIX=/usr/local/haproxy
5. mkdir /etc/haproxy
6.
7. vim /etc/haproxy/haproxy.cfg
8.
9. # This sample configuration makes extensive use of the ACLs. It requires
10. # HAProxy version 1.3.12 minimum.
11.
12. global #设定全局配置参数，属于进程级的配置，通常和操作系统配置有关
    log 127.0.0.1 local0 info #全局日志配置，local0是日志设备，info表示日志级别，分别有err,
    warning,info,debug可选。这个配置表示使用127.0.0.1上的rsyslog服务中的local0日志设备，记录日志等
    级为info
13.     maxconn 4096 #设定每个haproxy进程可接受的最大并发连接数，此选项等同于linux命令选项ul
    mit -n
14.     uid nobody #设定运行haproxy进程的用户和组，也可以使用用户和组的uid和gid的值
15.     gid nobody
16.     pidfile /var/run/haproxy.pid
17.     daemon #设置haproxy进程进入后台运行。这是推荐的运行模式
18.     nbproc 1 #设置haproxy启动时可创建的进程数，此参数要求将haproxy运行模式设置为daemon，
    默认只启动一个进程。根据使用经验，该值的设置应该小于服务器的cpu核数。
20.
21. defaults #默认参数配置。在此设置的参数，默认会自动引用到下面frontend/backend/listen部分中。
    如果某些参数属于公用的配置，在defaults中添加。如果在frontend/backend/listen中也配置了与defaults
    一样的参数。那么defaults部分参数对应的值自动被覆盖。
22.     mode http #设置haproxy实例默认的运行模式，有tcp/http可选值
23.         #tcp模式，客户端与服务端之间将建立一个全双工的连接，不会对七层报文做任何类型的检查，默认为tc
    p模式，经常用于ssl、ssh、smtp等应用
24.         #http模式，在此模式下，客户端请求在转发至后端服务器之前将会被深度分析，所有不与RFC格式兼容
```

的请求都会被拒绝

25. retries 3 #设置连接后端服务器的失败重试次数，如果连接失败的次数超过这里设置的值，haproxy会将对应的后端服务器标记为不可用，此参数也可在后面部分进行配置。

26. timeout connect 10s #设置成功连接到一台服务器的最长等待时间，默认单位是毫秒，但也可以使用其他的时间单位后缀

27. timeout client 20s #设置连接客户端发送数据时最长等待时间，默认单位是毫秒

28. timeout server 30s #连接服务器端回应客户端数据发送的最长等待时间，默认单位是毫秒

29. timeout check 5s #设置对后端服务器的检测超时时间，默认单位是毫秒

30.

31. frontend www #设置接收用户请求的前端虚拟节点。定义了一个www的前端虚拟节点

32. bind :80 #此选项只能在frontend和listen部分进行定义，用于定义一个或几个监听的套接字。格式

33. #bind [<address>:<port_range>] interface <interface>

34. #其中address为可选选项，其可以为主机名或者IP地址，如果将其设置为 * 或者 0.0.0.0，将监听当前系统的所有ipv4

35. mode http

36. option httplog #在默认情况下，haproxy日志是不记录http请求的。此选项启用日志记录

37. option forwardfor #如果后端服务器需要获得客户端的真实IP，就需要配置此选项

38. option httpclose #此选项表示在客户端和服务端完成一次连接请求后，haproxy将主动关闭此tcp请求。增加清能

39. log global #全局的日志配置。global表示在haproxy配置文件global部分中定义的log选项配置格式

40. default_backend htmpool #制定默认的后端服务器吃，也就是制定一组后端真实服务器。而这些真实服务器组将在backend段进行定义。这里的htmpool就是一个后端服务器组

41.

42. backend htmpool #用于设置集群后端服务集群的配置。也就是用来添加一组真是的服务器，以处理前端用户的请求。添加的真实服务器，类似LVS中的real server节点。

43. mode http

44. option redispatch #此参数御用cookie保持的环境中。在默认情况下，haproxy会将其请求的后端服务器的serverID插入cookie中，以保持会话session的持久性。而如果后端服务器出现故障，客户端的cookie是不会刷新的，这就会出现问题。如果设置此参数，就会将客户的请求强制定向到另外一台将抗的后端服务器上，以保证服务正常

45. option abortonclose #如果设置了此参数，可以在服务器负载很高的情况下，自动结束当前队列中处理时间比较长的连接。

46. balance roundrobin #此关键字用来定义负载均衡算法。

47. #roundrobin 基于权重进行轮叫调度的算法，在服务器的性能分部比较均匀时，这是一种最公平最合理的算法

48. #static-rr 基于权重进行轮叫的调度算法，不过此算法为静态方法，在运行时调整其服务器权重不会生效

49. #source 基于请求源ip的算法。此算法先对请求的源IP进行hash运算，然后将结果与后端服务器的权重总数相除后转发至某台匹配的后端服务器。这种方式可以使用同一个客户端IP的请求始终被装法到某特定的后端服务器

50. #leastconn 此算法会将新的连接请求转发到具有最少连接数目的后端服务器。在会话时间较长的场景中推荐使用此算法，例如数据库负载均衡等。此算法不适合会话短的环境，例如http

51. #此算法会对部分或整个URI进行hash运算，再经过与服务器的总权重相除，最后转发到某台匹配的后端服务器上

52. #uri_param 此算法会根据url路径中的参数进行转发，这样可保证在后端真实服务器数量不变时，同一个用户的请求始终分发到同一台机器上。

53. #hdr(<name>) 此算法根据http头进行转发，如果指定的http头名称不存在，则使用roundrobin算法进行策略转发

54. cookie SERVERID #表示允许向cookie插入SERVERI，每台服务器的SERVERID可在下面的server关键字中使用cookie关键字定义

55. option httpchk GET /index.php # option httpchk 选项表示启用http的服务器状态监测功能。以保证在后端backend中某个节点不能服务时，把从frontend端进来的客户端请求分配至backend的其他健康节

点上。

```

56. #格式 option httpckh <method> <uri> <version>
57. #method 表示http请求的方式，有OPTIONS/GET/HEAD几种方式。一般的健康检查可以采用HEAD方
式进行，而不是采用GET方式，因为HEAD方式没有数据放回，仅检查response的head是不是200。相对于
get, head更快
58. #uri 表示要检测的uri地址，通过执行此uri，获得后端服务器的运行状态。整成200
59. #version 指定心跳检测时的http版本号
60. server web1 10.200.34.181:80 cookie server1 weight 6 check inter 2000 rise 2 fall 3
61. #server 用来定义多台后端真实服务器，不能用户defaults和frontend部分
62. #server <name> <address>[:port] [param*]
63. #name 后端真实服务器指定一个内部名称，随便定义一个即可
64. #address 后端真实服务器的ip地址或者主机名
65. #port 指定连接请求发往真实服务器时的目标端口。在未设定时，将使用客户端请求的同一端口
66. #param 为后端服务器设定的一系列参数，常用的
67. #check 表示启用对此后端服务器执行健康状态的检查
68. #inter 设置将抗状态检查的时间间隔，单位为毫秒
69. #rise 设置从故障状态转换至此正常状态需要成功检查的次数。rise 2 表示2此检查正确就认为此服务器可
用
70. #fall 设置后端服务器从正常状态转换为不可用状态需要检查的次数。fall 3 表示检查3次失败就认定此服
务器不可用
71. #cookie 为指定的后端服务器设定cookie值，此处指定的值将在请求入站时被检查，第一次为此值挑选的
后端服务器将在后续的请求中一致被选中，其目的在于实现持久连接的功能。上面的cookie server1表示we
b1的serverid为server1。
72. #weight 设置后端真实服务器的权重，默认为1，最大值256.设置为0表示不参与负载均衡。
73. #backup 设置后端真实服务器的备份服务器，仅仅在后端所有真实服务器均不可用的情况下才启用
74. server web2 10.200.34.182:8080 cookie server2 weight 6 check inter 2000 rise 2 fall 3
75.
76. listen admin_stats #此部分中frontend部分和backend部分的结合体。定义了一个名为admin_stats的
实例，其实就是定义了一个haproxy的监控页面
77. bind 0.0.0.0:9188
78. mode http
79. log 127.0.0.1 local0 err
80. stats refresh 30s #设置haproxy监控统计页面的自动刷新时间
81. stats uri /haproxy-status #设置haproxy监控统计页面的url路径。此设置，可以通过http://ip:918
/haproxy-status 查看
82. stats realm welcome login\ Haproxy #设置登录haproxy统计页面时密码框上的文本提示信息
83. stats auth admin:admin~!@ #设置登录haproxy统计页面的用户名和密码。用户名和密码通过冒号
分隔。可为监控页面设置多个用户名密码，每行一个
84. stats hide-version #隐藏统计页面上的haproxy版本信息
85. stats admin if TRUE #可以在监控页面上手工启用或禁用后端真实服务器。

```

日志配置

```

1. [root@c2 pkg]# rpm -q rsyslog
2. rsyslog-5.8.10-8.el6.x86_64
3.
4. vim /etc/rsyslog.d/haproxy.conf
5.
6. $ModLoad imudp #imudp是模块名，支持UDP协议
7. $UDPServerRun 514 #表示允许514端口接受使用UDP和TCP协议转发过来的日志，而rsyslog在默认情
况下，正事在514端口监听UDP。其实也可以将上面haproxy.conf文件的内容直接写到/etc/rsyslog.conf文
件中。然后，在需要修改/etc/sysconfig/rsyslog文件，修改内容：
8. SYSLOGD_OPTIONS="-c 2 -r -m 0"

```

```

9. #其中, -r表示接受远程日志。最后重启rsyslog服务即可完成配置
10. local3.* /usr/local/haproxy/logs/haproxy.log
11. local0.* /usr/local/haproxy/logs/haproxy.log
12.
13. service rsyslog restart

```

通过haproxy的acl规则实现智能负载均衡， haproxy通过acl规则完成两种主要功能：

1. 通过设置的 acl 规则检查客户端请求是否合法。如果符合， 将放行；如果不合法，则直接中断请求。
2. 符合acl规则要求的将被提交到后端的backend服务器集群，进而实现基于acl规则的负载均衡

haproxy中的acl规则经常使用在frontend段中， 格式：

acl 自定义的acl名称 acl方法 -i [匹配的路径或者文件]

acl：是关键字， 表示定义acl规则的开始。后面需要跟上自定义的acl名称

acl方法：这个字段用来定义实现acl的方法， haproxy定义了很多acl方法， 经常使用的有
hdr_reg(host)、 hdr_dom(host)、 url_sub、 url_dir、 path_beg、 path_end等

-i：表示不区分大小写， 后面需要跟上匹配的路径或文件或正则表达式

与acl规则一起使用的haproxy参数还有use_backend,use_backend后面需要跟上一个backend实例名， 表示在满足acl规则后去请求哪个backend实例， 与use_backend对应的还有
default_backend参数， 它表示在没有满足acl条件的时候默认使用哪个后端backend

acl 常见规则举例

```

1. acl www_policy hdr_reg(host) -i ^www.z.cn|z.cn #如果客户端以www.z.cn或z.cn开头的
   域名发送请求时，则此规则返回true
2. acl bbs_policy hdr_dom(host) -i bbs.z.cn
3. acl url_policy url_sub -i buy_sid= #客户端请求的url中包含了buy_sid=字符串时，返回tr
   ue
4.
5. use_backend server_www if www_policy #当满足www_policy规则时，那么haproxy会将
   用户的请求直接发往server_www
6. use_backend server_app if url_policy
7. use_backend server_bbs if bbs_policy
8. default_backend server_cache #当不满足任何一个acl规则时，haproxy就会把请求发往由default
   _backend选项指定的server_cache这个后端backend

```

再来个例子

```

1. acl url_static path_end .gif .png .jpg .css .js #如果客户端在请求的url中以.gif等结尾，返回t
   rue
2. acl host_www hdr_beg(host) -i www #如果客户端以www开头的域名发送请求时则返回tru
   e

```

```

3. acl host_static hdr_beg(host) -i img. video. download. ftp. #以img.等开头的域名发送请求返回true
4.
5. use_backend static if host_static || host_www url_static #表示符合规则后要调度到哪个backend
6. use_backend www if host_www
7. default_backend server_cache

```

通过haproxy的acl规则配置虚拟主机

需求，通过haproxy，调度不同的域名

```

1. haproxy
2.   ├── bbs.tb.com或tb.com
3.   ├── blog.tb.com
4.   ├── default
5.   └── www.tb.com
6.
7. vim haproxy.cfg
8.
9. # This sample configuration makes extensive use of the ACLs. It requires
10. # HAProxy version 1.3.12 minimum.
11.
12. global
13.   log 127.0.0.1 local0 info
14.   maxconn 4096
15.   uid nobody
16.   gid nobody
17.   pidfile /var/run/haproxy.pid
18.   daemon
19.   nbproc 1
20.
21. defaults
22.   mode http
23.   retries 3
24.   timeout connect 5s
25.   timeout client 30s
26.   timeout server 30s
27.   timeout check 2s
28.
29. listen admin_stats
30.   bind 0.0.0.0:19188
31.   mode http
32.   log 127.0.0.1 local0 err
33.   stats refresh 30s
34.   stats uri /haproxy-status
35.   stats realm welcome login\ Haproxy
36.   stats auth admin:admin~!@
37.   stats auth admin1:xxxxxxxx
38.   stats hide-version
39.   stats admin if TRUE
40.
41. frontend www

```

```
42. bind      :80
43. mode       http
44. option     httplog
45. option     forwardfor
46. log        global
47.
48. acl host_www hdr_reg(host) -i ^(.+\.tb\.com|tb\.com)
49. acl host_bbs hdr_dom(host) -i bbs.tb.com
50. acl host_blog hdr_beg(host) -i blog.
51.
52. use_backend server_www if host_www
53. use_backend server_bbs if host_bbs
54. use_backend server_log if host_blog
55. default_backend server_default
56.
57. backend server_default
58.   mode      http
59.   option    redispatch
60.   option    abortonclose
61.   balance   roundrobin
62.   cookie    SERVERID
63.   option    httpchk GET /check_status.html
64.   server    default1 192.168.88.90:8000 cookie default1 weight 3 check inter 2000 rise 2 fall
65.           1 3
66.           server default2 192.168.88.91:8000 cookie default2 weight 3 check inter 2000 rise 2 fall
67.           1 3
68.
69. backend server_default
70.   mode      http
71.   option    redispatch
72.   option    abortonclose
73.   balance   source
74.   cookie    SERVERID
75.   option    httpchk GET /check_status.jsp
76.   server    www1 192.168.88.80:80 cookie www1 weight 6 check inter 2000 rise 2 fall 3
77.   server    www2 192.168.88.81:80 cookie www2 weight 6 check inter 2000 rise 2 fall 3
78.   server    www3 192.168.88.82:80 cookie www3 weight 6 check inter 2000 rise 2 fall 3
79.
80. backend server_bbs
81.   mode      http
82.   option    redispatch
83.   option    abortonclose
84.   balance   source
85.   cookie    SERVERID
86.   option    httpchk GET /check_status.php
87.   server    bbs1 192.168.88.83:8080 cookie bbs1 weight 8 check inter 2000 rise 2 fall 3
88.   server    bbs2 192.168.88.84:8090 cookie bbs2 weight 8 check inter 2000 rise 2 fall 3
89.
90. backend server_blog
91.   mode      http
92.   option    redispatch
93.   option    abortonclose
94.   balance   roundrobin
95.   cookie    SERVERID
```

```

94.    option httpchk GET /check_blog.php
95.    server blog1 192.168.88.85:80 cookie blog1 weight 8 check inter 2000 rise 2 fall 3
96.    server blog2 192.168.88.86:80 cookie blog2 weight 8 check inter 2000 rise 2 fall 3

```

启动haproxy

- [root@c1 haproxy]# /usr/local/sbin/haproxy -f /etc/haproxy/haproxy.cfg

常用参数

- v : 显示当前版本信息, "-vv"显示已知的创建选项
- d : 表示让进程运行在debug模式, "-db"表示禁用后台模式, 让程序在前台运行
- D : 让程序以daemon模式运行, 此选项也可以在haproxy配置文件中设置
- q : 表示安静模式, 程序运行部输出任何信息
- c : 对haproxy配置文件进行语法检查。此参数非常有用。如果配置文件错误, 会输出对应的错误位置和错误信息
- n : 设置最大并发连接总量
- m : 限制可用的内存大小, 以MB为单位
- N : 设置默认的连接数
- p : 设置haproxy的pid文件路径
- de : 不使用epoll模型
- ds : 不使用speculative epoll
- dp : 不使用poll模型
- sf : 程序启动后向pid文件里的进程发送FINISH信号, 这个参数需要放在命令的最后
- st : 程序启动后向pid文件里的进程发送TERMINATE信号, 这个参数放在命令行的最后, 进场用于重启haproxy进程

关闭haproxy

- killall -9 haproxy

平滑重启

- /usr/local/sbin/haproxy -f /etc/haproxy/haproxy.cfg -st `/var/run/haproxy.pid`

haproxy的启动脚本

```

1. #!/bin/sh
2. # confit: /usr/local/haproxy/conf/haproxy.cfg
3. # pidfile: /usr/local/haproxy/logs/haproxy.pid
4.
5. # source function library.
6. . /etc/rc.d/init.d/functions
7.

```

```

8. # source networking configuration.
9. . /etc/sysconfig/network
10.
11. # check that networking is up.
12. [ "$NETWORKING" = "no" ] && exit 0
13. config="/usr/local/haproxy/conf/haproxy.cfg"
14. exec="/usr/local/haproxy/sbin/haproxy"
15. prog=$(basename $exec)
16.
17. [ -e /etc/sysconfig/$prog ] && . /etc/sysconfig/$prog
18.
19. lockfile=/var/lock/subsys/haproxy
20.
21. check(){
22.     $exec -c -V -f $config
23. }
24. start() {
25.     $exec -c -q -f $config
26.     if [ $? -ne 0 ]; then
27.         echo "Errors in configuration file, check with $prog check."
28.         return 1
29.     fi
30.
31.     echo -n $"Starting $prog: "
32.     # start it up here, usually something like "daemon $exec"
33.     daemon $exec -D -f $config -p /var/run/$prog.pid
34.     retval=$?
35.     echo
36.     [ $retval -eq 0 ] && touch $lockfile
37.     return $retval
38. }
39.
40. stop() {
41.     echo -n $"Stopping $prog: "
42.     # stop it here, often "killproc $prog"
43.     killproc $prog
44.     retval=$?
45.     echo
46.     [ $retval -eq 0 ] && rm -f $lockfile
47.     return $retval
48. }
49.
50. restart() {
51.     $exec -c -q -f $config
52.     if [ $? -ne 0 ]; then
53.         echo "Errors in configuration file, check with $prog check."
54.         return 1
55.     fi
56.     stop
57.     start
58. }
59.
60. reload() {
61.     $exec -c -q -f $config

```



```

62. if [ $? -ne 0 ]; then
63.     echo "Errors in configuration file, check with $prog check."
64.     return 1
65. fi
66. echo -n $"Reloading $prog: "
67. $exec -D -f $config -p /var/run/$prog.pid -sf $(cat /var/run/$prog.pid)
68. retval=$?
69. echo
70. return $retval
71. }
72.
73. force_reload() {
74.     restart
75. }
76.
77. fdr_status() {
78.     status $prog
79. }
80.
81. case "$1" in
82.     start|stop|restart|reload)
83.         $1
84.         ;;
85.     force-reload)
86.         force_reload
87.         ;;
88.     checkconfig)
89.         check
90.         ;;
91.     status)
92.         fdr_status
93.         ;;
94.     condrestart|try-restart)
95.         [ ! -f $lockfile ] || restart
96.         ;;
97.     *)
98.         echo $"Usage: $0 {start|stop|status|checkconfig|restart|try-restart|reload|force-reloa
d}"
99.         exit 2
100. esac

```

haproxy 的监控功能

web已在配置文件中指定，直接访问即可

本文出自若我若鱼,原文链接[haproxy 的安装和配置](#),转载请遵照CC BY-NC-SA协议规定.

[添加新评论](#)

称呼邮箱网站 <http://example>提交评论

© 2007-2016 若我若鱼. 黑ICP备15005488号. 网站地图.



本作品采用知识共享署名-非商业性使用-相同方式共享 4.0 国际许可协议进行许可。

