



原创

# HAProxy入门及常用配置模拟测试



Doumadouble

2018-07-10 16:08:44 20202人阅读 1人评论

## HAProxy简介

HAProxy是一个使用C语言编写的，提供负载均衡，以及基于TCP（伪四层）和HTTP（七层）的应用程序代理。

HAProxy特别适用于那些负载大的web站点，这些站点通常又需要会话保持或七层处理。HAProxy完全可以支持数以万计的并发连接。并且它的运行模式使得它可以很简单安全的整合进您当前的架构中，同时可以保护你的web服务器不被暴露到网络上。

HAProxy实现了一种事件驱动, 单一进程模型, 此模型支持非常大的并发连接数。多进程或多线程模型受内存限制、系统调度器限制以及无处不在的锁限制，很少能处理数千并发连接。事件驱动模型因为在有更好的资源和时间管理的用户空间(User-Space) 实现所有这些任务，所以没有这些问题。

## HAProxy与LVS对比

1.都是负载均衡的实现产品，但是LVS是基于Linux操作系统的软负载均衡；HAProxy是基于第三方应用的软负载均衡。

2.LVS是基于四层的IP负载均衡方案；HAProxy是同时具有四层和七层的，可以提供TCP和HTTP应用的负载均衡方案。

3.LVS工作在OSI参考模型的四层，所以其状态监控功能比较简陋，HAProxy在状态监控方面更为优秀，可以支持基于端口，URL，脚本等多种状态监测方法。

4. HAProxy在整体性能上明显低于四层负载的LVS；LVS技术由于Linux系统内核的支持有着接近于硬件设备的负载能力。

## 部署及配置文件介绍

### 安装部署



在线客服



## yum安装

安装haproxy

```
yum install haproxy -y
```

## 源码编译安装

1.安装开发环境

```
yum groupinstall "development tools" -y
```

2.下载源码包

```
wget https://src.fedoraproject.org/repo/pkgs/haproxy/haproxy-1.8.12.tar.gz/sha512/2b782a54988cc88d1af0e5f
```

3.解压并安装

```
tar -xvf haproxy-1.8.12.tar.gz
cd haproxy-1.8.12

#获取内核版本信息
uname -r

# TARGET=linux310, 内核版本, 使用uname -r来查看
# ARCH指明系统的位数
# PREFIX指明安装的路径
make TARGET=linux310 ARCH=x86_64 PREFIX=/usr/local/haproxy
make install PREFIX=/usr/local/haproxy
```

4.准备配置文件

```
#创建对应目录
mkdir /usr/local/haproxy/conf
#创建配置文件
cd /usr/local/haproxy/conf
vim haproxy.cfg

#这里复制一份yum安装的配置文件就好, 也可以自己写一份
```

5.添加用户

```
useradd -r -s /usr/sbin/nologin haproxy
```

6.启动haproxy

```
# -f指明配置文件, 配置文件一定要写对, 否则无法启动haproxy
/usr/local/haproxy/sbin/haproxy -f /usr/local/haproxy/conf/haproxy.cfg
```

## 配置文件介绍

### 配置文件大体分类

**全局配置段 (global) :** (建议不做改动, 会自动调整)

- 1.进程与安全配置相关的参数设定 ;
- 2.性能参数配置段 ;
- 3.Debug参数配置段 ;
- 4.用户列表段 ;
- 5.peer段

**代理配置段 (proxies) :**



在线  
客服



段，backend段和listen段进行配置的参数同时在default也有相应参数配置的话，default段的配置会被覆盖掉。

2.**frontend段**：本段负责配置接收用户请求的虚拟前端节点。类似于nginx的server{}部分。

3.**backend段**：本段用于设置集群后端服务器集群的配置，也就是用来定义一组真实服务器，来处理用户发出的请求。添加的真实服务器类似于LVS中的real-server，相当于nginx中的 upstream {}段

4.**listen段**：同时用来负责配置前端和happ后端。

## 配置文件详解

### global配置段

```
#-----
# Example configuration for a possible web application. See the
# full configuration options online.
#
# http://haproxy.lwt.eu/download/1.4/doc/configuration.txt
#
#-----
#
#-----
# Global settings
#-----
global
    # to have these messages end up in /var/log/haproxy.log you will
    # need to:
    #
    # 1) configure syslog to accept network log events. This is done
    #    by adding the '-r' option to the SYSLOGD_OPTIONS in
    #    /etc/sysconfig/syslog
    #
    # 2) configure local2 events to go to the /var/log/haproxy.log
    #    file. A line like the following can be added to
    #    /etc/sysconfig/syslog
    #
    #    local2.*                /var/log/haproxy.log
    #
    log                127.0.0.1 local2

    chroot             /var/lib/haproxy
    pidfile             /var/run/haproxy.pid
    maxconn             4000
    user               haproxy
    group              haproxy
    daemon

    # turn on stats unix socket
    stats socket /var/lib/haproxy/stats
```



#### #常用配置项

**log**：全局日志文件配置条目，local2表示日志设备，最多可定义两个

**nbproc**：要启动的haproxy的进程数量

**chroot**：**chroot** 切换根目录，将haproxy都运行在/var/lib/haproxy 这样做是为了增加haproxy的安全。

**pidfile**：指定haproxy的进程pid文件，启动进程的用户必须要有访问该文件的权限。

**maxconn**：设定每个haproxy进程可以接受的最大并发连接数。

**user**和**group**：指定运行haproxy的用户和组

**daemon**：设置haproxy以后台运行的方式运行。

**maxconnrate**：每个进程每秒最大处理的连接数量。

**maxse\*\*\*ate**：每个进程每秒可以创建的最大会话速率。

**maxsslconn**：设定每个haproxy进程所能接受的ssl最大并发连接数。

### default配置段

在线  
客服



```
defaults
    mode                http
    log                 global
    option              httplog
    option              dontlognull
    option http-server-close
    option forwardfor   except 127.0.0.0/8
    option              redispatch
    retries             3
    timeout http-request 10s
    timeout queue       1m
    timeout connect     10s
    timeout client      1m
    timeout server      1m
    timeout http-keep-alive 10s
    timeout check       10s
    maxconn             3000

#-----@51CTO博客
```

#### #常用配置项

mode: 设置haproxy实例默认的运行模式, 默认是http, 支持tcp, http可选值

tcp模式: 在该模式之下, 客户端会与服务器端建立一个全双工连接, 不对七层报文做检查, 常用语ssl,

http模式: 客户端在请求转发到后端服务器之前会被分析。

log global: 继承全局日志

option dontlognull: 保证HAProxy不记录上级负载均衡发送过来的用于检测状态没有数据的心跳包。

option http-server-close: 客户端与服务器端在完成一次连接请求之后, HAProxy会主动关闭该TCP连接, 有B  
xforwardfor: 由于haproxy工作在反向代理方向, 因此后端的真实服务器可能无法获取真实的请求端ip, 使用xfo  
redispatch: 是否允许在session 失败后重新分配。

retries: 设置连接后端服务器的失败重试次数, 连接失败的次数如果超过这里设置的值, haproxy将会将对应的后

timeout connect: 成功连接到一台服务器的最长等待时间, 默认为毫秒, 可以换用其他单位。

timeout client: 连接客户端发送数据的最长等待时间, 默认毫秒, 可修改。

timeout server: 服务器端回应客户端数据发送的最长等待时间, 默认毫秒, 可以修改。

timeout check: 设置对后端服务器的检测超时时间, 默认毫秒, 可以修改



## 重点配置参数

1.bind: 配置监听套接字, 不能有默认值, 不能用在backend中。

bind :80,:443 同时监听2个端口 (之间不能有空格, 监听端口要重启服务)

bind ip:port,ip:port

bind /var/\*\*\*\*.sock 使用套接字文件

2.balance: 后端服务器组内的调度算法

roundrobin: 轮询, 依次访问每一个后端ip (短连接和无状态的连接推荐使用rr算法)

server options: weight # 支持配置权重

动态算法: 支持权重的运行时调整, 支持慢启动; 每个后端中最多支持4095个server;

static-rr: 静态算法: 不支持权重的运行时调整及慢启动; 后端主机数量无上限;



在线  
客服



配置文件：

```
1 haproxy x 2 httpd x 3 httpd x +
stats socket /var/lib/haproxy/stats

#-----
# common defaults that all the 'listen' and 'backend' sections will
# use if not designated in their block
#-----
defaults
    mode                http
    log                 global
    option              httplog
    option              dontlognull
    option http-server-close
    option forwardfor   except 127.0.0.0/8
    option redispatch
    retries             3
    timeout http-request 10s
    timeout queue       1m
    timeout connect     10s
    timeout client      1m
    timeout server      1m
    timeout http-keep-alive 10s
    timeout check       10s
    maxconn             3000

#-----
# main frontend which proxys to the backends
#-----
frontend main
    bind :80
    mode http
    default_backend webserver

backend webserver
    balance roundrobin
    server webserver 192.168.99.131:80 check
    server webserver 192.168.99.135:80 check

73,0-1 Bot
@51CTO博客
```

请求效果

```
1 haproxy x 2 httpd x 3 httpd x +
[root@localhost haproxy]# for i in {1..10};do curl 192.168.99.130;done
web2
web1
web2
web1
web2
web1
web2
web1
web2
web1
[root@localhost haproxy]#
```

加权轮询

在线客服

```
stats socket /var/lib/haproxy/stats

#-----
# common defaults that all the 'listen' and 'backend' sections will
# use if not designated in their block
#-----
defaults
    mode                http
    log                 global
    option              httplog
    option              dontlognull
    option http-server-close
    option forwardfor   except 127.0.0.0/8
    option              redispatch
    retries              3
    timeout http-request 10s
    timeout queue        1m
    timeout connect     10s
    timeout client       1m
    timeout server       1m
    timeout http-keep-alive 10s
    timeout check        10s
    maxconn             3000

#-----
# main frontend which proxys to the backends
#-----
frontend main
    bind :80
    mode http
    default_backend webserver

backend webserver
    balance roundrobin
    server webserver 192.168.99.131:80 check weight 3
    server webserver 192.168.99.135:80 check
```

默认权重为1；其中一台设置为3

效果

```
[root@localhost haproxy]# vim haproxy.cfg
[root@localhost haproxy]# for i in {1..10};do curl 192.168.99.130;done
web1
web2
web1
web1
web1
web2
web1
web1
web1
web2
[root@localhost haproxy]#
```



- leastconn：推荐使用在具有较长会话（长连接，有状态）的场景中，例如MySQL、LDAP等；
- first：先到先得，根据服务器在列表中的位置，自上而下进行调度；前面服务器的连接数达到上限，新请求才会分派
- source：源地址hash，本质上是source ip，将同一个ip的请求发往同一服务器，用于保持会话。（snat模式与源地址hash结合使用，效果更佳）
  - 除权取余法（一旦服务器组发生变化会产生巨大影响）：map-based
  - 一致性哈希（最佳，但是由于做大量运算会影响性能）：consistent
- uri：对URI的左半部分做hash计算，并由服务器总权重相除以后派发至某挑出的服务器；有助于提升缓存命中率。

在线客服

<scheme>://<user>:<password>@<host>:<port>/<path>;<params>?<query>

右半部分：<path>:<params>

uri 算法测试

1 haproxy2 httpd3 httpd

```
#-----
# common defaults that all the 'listen' and 'backend' sections will
# use if not designated in their block
#-----
defaults
    mode                http
    log                 global
    option              httplog
    option              dontlognull
    option http-server-close
    option forwardfor   except 127.0.0.0/8
    option redispatch
    retries             3
    timeout http-request 10s
    timeout queue       1m
    timeout connect     10s
    timeout client      1m
    timeout server      1m
    timeout http-keep-alive 10s
    timeout check       10s
    maxconn             3000

#-----
# main frontend which proxys to the backends
#-----
frontend main
    bind :80
    mode http
    default_backend webserver

backend webserver
    balance uri
    hash-type consistent
    server webserver 192.168.99.131:80 check weight 3
    server webserver 192.168.99.135:80 check
```

74,0-1Bot  
@51CTO博客

指明使用uri算法和一致性hash

效果

1 haproxy2 httpd3 httpd

```
[root@localhost haproxy]# for i in {1..10};do curl 192.168.99.130/test1.html;done
this is test1 @web1
this is test1 @web1
this is test1 @web1
this is test1 @web1
this is test1 @web1
this is test1 @web1
this is test1 @web1
this is test1 @web1
this is test1 @web1
this is test1 @web1
[root@localhost haproxy]# for i in {1..10};do curl 192.168.99.130/test2.html;done
this is test2 @web1
this is test2 @web1
this is test2 @web1
this is test2 @web1
this is test2 @web1
this is test2 @web1
this is test2 @web1
this is test2 @web1
this is test2 @web1
this is test2 @web1
[root@localhost haproxy]# for i in {1..10};do curl 192.168.99.130/test3.html;done
this is test3 @web2
this is test3 @web2
this is test3 @web2
this is test3 @web2
this is test3 @web2
this is test3 @web2
this is test3 @web2
this is test3 @web2
this is test3 @web2
this is test3 @web2
[root@localhost haproxy]# for i in {1..10};do curl 192.168.99.130/test3.html;done
```

74,0-1Bot  
@51CTO博客

请求同一页面全部被调度到同一服务器



在线客服

url\_param: 对用户请求的uri的<params>部分中的参数的值作hash计算，并由服务器总权重相除以后派发至某挑

hdr(<name>): 对于每个http请求，此处由<name>指定的http首部将会被取出做hash计算； 并由服务器总权重相

hdr(Cookie)

```
rdp-cookie
rdp-cookie(<name>)
```

hash-type: 哈希算法

hash-type <method> <function> <modifier>

map-based: 除权取余法，哈希数据结构是静态的数组；

consistent: 一致性哈希，哈希数据结构是一个树；

<function> is the hash function to be used : 哈希函数

sdbm

djb2

wt6

hdr算法匹配报文字段请求（这里匹配的是浏览器）

1 haproxy

2 httpd

3 httpd

+

```
#-----
# common defaults that all the 'listen' and 'backend' sections will
# use if not designated in their block
#-----
defaults
    mode                http
    log                 global
    option              httplog
    option              dontlognull
    option http-server-close
    option forwardfor   except 127.0.0.0/8
    option redispatch
    retries             3
    timeout http-request 10s
    timeout queue       1m
    timeout connect     10s
    timeout client      1m
    timeout server      1m
    timeout http-keep-alive 10s
    timeout check       10s
    maxconn             3000

#-----
# main frontend which proxys to the backends
#-----
frontend main
    bind :80
    mode http
    default_backend webserver

backend webserver
    balance hdr(User-Agent)
    hash-type consistent
    server webserver 192.168.99.131:80 check weight 3
    server webserver 192.168.99.135:80 check
```

匹配user-agent字段

74,0-1 Bot

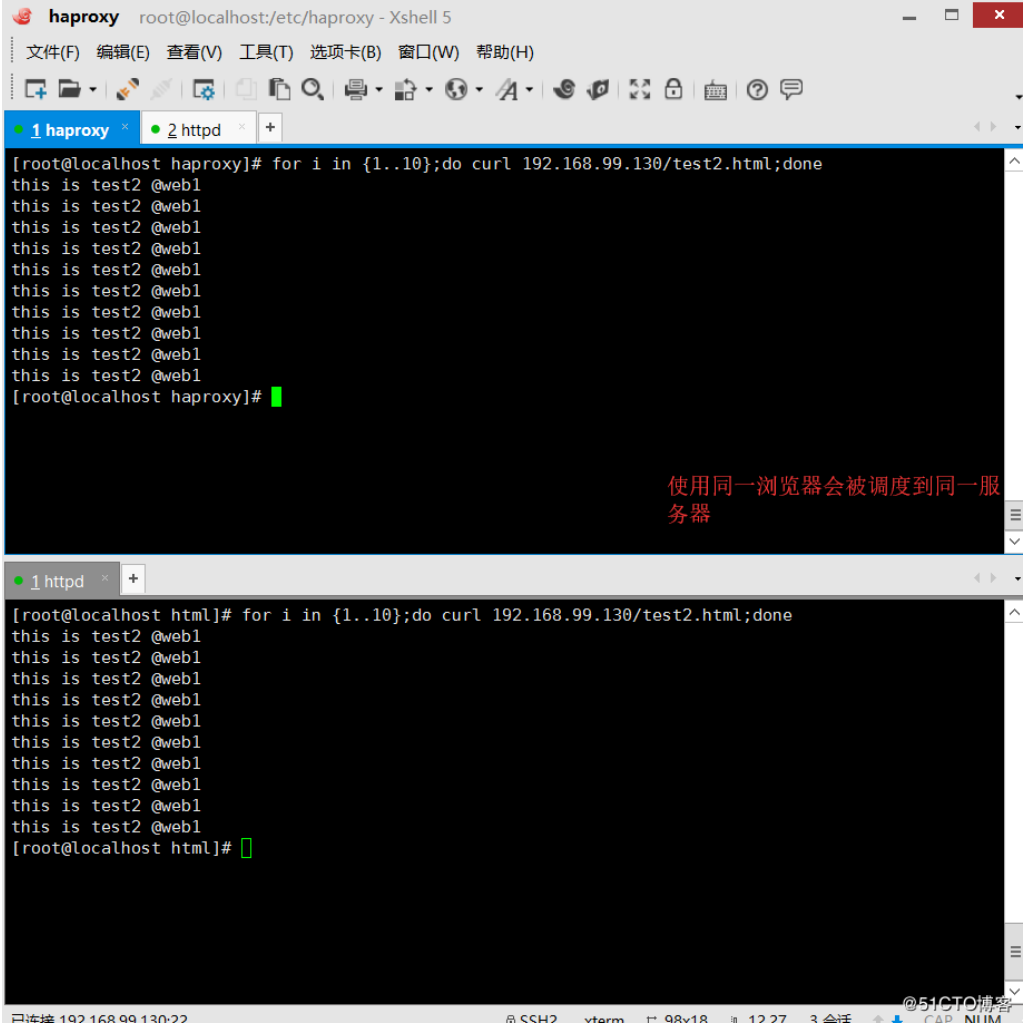
已连接 192.168.99.130:22, SSH2 xterm 98x39 38,1 3 会话 @51CTO博客



在线  
客服



效果



3.default\_backend：设置默认的后端主机组，在frontend中定义

4.server的配置，定义后端主机的各个服务器及选项。同一后端主机可以被多个backend引用。

- name：服务器在haproxy上的内部名称，主要出现在日志和警告中；
- address：服务器地址，可以使用主机名替代（重要）
- maxconn<maxconn>：当前server的最大并发连接数；
- backlog<backlog>：当前server的连接数达到上限后的后援队列长度；
- backup：设定当前server为备用服务器；
- check：对当前server做健康状态检测；默认是tcp的检测，要关注状态变化

- addr：检测时使用的IP地址；
- port：针对此端口进行检测；
- inter<delay>：连续两次检测之间的时间间隔，默认为2000ms；
- rise<count>：连续多少次检测结果为“成功”才标记服务器为可用；默认
- fail<count>：连续多少次检测结果为“失败”才标记服务器为不可用；

注意：option httpchk, "smtpchk", "mysql-check", "pgsql-check" and "ssl-hello-chk" 用



- cookie<value>：为当前server指定其cookie值，用于实现基于cookie的会话黏性；
- disabled：标记为不可用；一般用来做发布
- on-error<mode>：后端服务故障时的行动策略；

- fastinter: force fastinter
- fail-check: simulate a failed check, also forces fasti
- sudden-death: simulate a pre-fatal failed health check
- check will mark a server down, forces fastinter

redir <prefix>: 将发往此server的所有GET和HEAD类的请求重定向至指定的URL;

```
#-----
defaults
    mode                http
    log                 global
    option              httplog
    option              dontlognull
    option http-server-close
    option forwardfor   except 127.0.0.0/8
    option              redispatch
    retries             3
    timeout http-request 10s
    timeout queue       1m
    timeout connect     10s
    timeout client      1m
    timeout server      1m
    timeout http-keep-alive 10s
    timeout check       10s
    maxconn             3000

#-----
# main frontend which proxys to the backends
#-----

frontend main
    bind :80
    mode http
    default_backend webserver

backend webserver
    balance hdr(User-Agent)
    hash-type consistent
    server webserver 192.168.99.131:80 check disable
    server webserver 192.168.99.135:80 check redir https://www.baidu.com
```

重定向到百度，  
可以自定义重定向地址

weight <weight>: 权重，默认为1。

5.基于cookie做会话绑定



在线客服

1 haproxy

2 httpd

3 httpd

```
#-----
# common defaults that all the 'listen' and 'backend' sections will
# use if not designated in their block
#-----
defaults
    mode                http
    log                 global
    option              httplog
    option              dontlognull
    option http-server-close
    option forwardfor   except 127.0.0.0/8
    option redispatch
    retries             3
    timeout http-request 10s
    timeout queue       1m
    timeout connect     10s
    timeout client      1m
    timeout server      1m
    timeout http-keep-alive 10s
    timeout check       10s
    maxconn             3000

#-----
# main frontend which proxys to the backends
#-----
frontend main
    bind :80
    mode http
    default_backend webserver

backend webserver
    balance roundrobin
    cookie WEBSERVER insert nocache indirect
    server webserver1 192.168.99.131:80 check cookie webserver1
    server webserver2 192.168.99.135:80 check cookie webserver2
```

74,0-1 Bot @51CTO博客

启用cookie 并且只是插入到没有缓存和重定向的请求

51CTO博客2.0-原创IT技

192.168.99.130

192.168.99.130

应用 Google 我的关注 - 个人中心 Oracle教程™ - 一个 Python教程 - 廖雪峰

web1

Elements Console Sources Network Performance Memory Application Security Audits >>

View: Group by frame Preserve log Disable cache Offline Online

Filter Hide data URLs All XHR JS CSS Img Media Font Doc WS Manifest Other

10 ms 20 ms 30 ms 40 ms 50 ms 60 ms 70 ms 80 ms 90 ms 100 ms 110

Name

192.168.99.130

bca9374b-8fac-40ca-9518-5c5...

Headers

Preview

Response

Cookies

Timing

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,\*/\*;q=0.8

Accept-Encoding: gzip, deflate

Accept-Language: zh-CN,zh;q=0.9

Cache-Control: no-cache

Connection: keep-alive

Cookie: wp-settings-1=libraryContent%3Dupload; wp-settings-time-1=1529989894; WEBSERVER=webserver1

DNT: 1

Host: 192.168.99.130

Pragma: no-cache

Upgrade-Insecure-Requests: 1

User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/66.0.3359.139 Safari/537.36

2 requests | 244 B transferred | Fi...

Console

top

Filter

Default levels

Group similar

@51CTO博客



在线客服

对后端服务器做http协议的健康状态检测:

```
option httpchk
option httpchk <method> <uri>
option httpchk <method> <uri> <version>
```

定义基于http协议的7层健康状态检测机制

## 7.接口，统计配置

## 配置状态页

[illegible]

[192.168.99.130/haproxy/status](#)

[应用](#)
[Google](#)
[我的关注](#)
[个人中心](#)
[Oracle教程™](#)
[一个](#)
[Python教程](#)
[廖雪松](#)
[入门, 第 1 部分](#)
[MaHua 在线markd](#)

## HAProxy version 1.5.18, released 2016/05/10

### Statistics Report for pid 17042

#### > General process information

```

pid = 17042 (process #1, nbproc = 1)
uptime = 6d 19h 10m1s
system limits: memmax = unlimited, ulimit-n = 8033
maxsock = 8033, maxconn = 4000, maxpipes = 0
current tasks = 1, current pipes = 0/0, conn rate = 1/sec
Running tasks = 1/7, 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

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](#)

main		Queue				Session rate				Sessions				Total				Bytes				Denied				Errors				Warnings				Status				Server			
Frontend	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Tot	La	St	La	St	In	Out	Req	Res	Req	Res	Conn	Resp	Retr	Redis	St	La	Chk	Wght	Act	Bck	Chk	Dwn	Dwtime	Thrtle				
	1	1	-	1	1	3 000	1											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			

webserver		Queue				Session rate				Sessions				Total				Bytes				Denied				Errors				Warnings				Status				Server			
webserver1	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Tot	La	St	La	St	In	Out	Req	Res	Req	Res	Conn	Resp	Retr	Redis	St	La	Chk	Wght	Act	Bck	Chk	Dwn	Dwtime	Thrtle				
webserver1	0	0	-	0	0	0	0	0	0	0	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
webserver2	0	0	-	0	0	0	0	0	0	0	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
Backend	0	0	-	0	0	0	0	0	0	0	0	300	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			

@51CTO精英

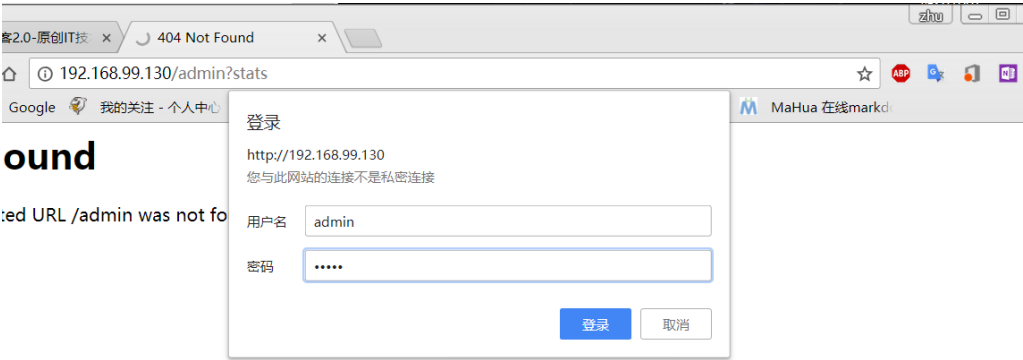
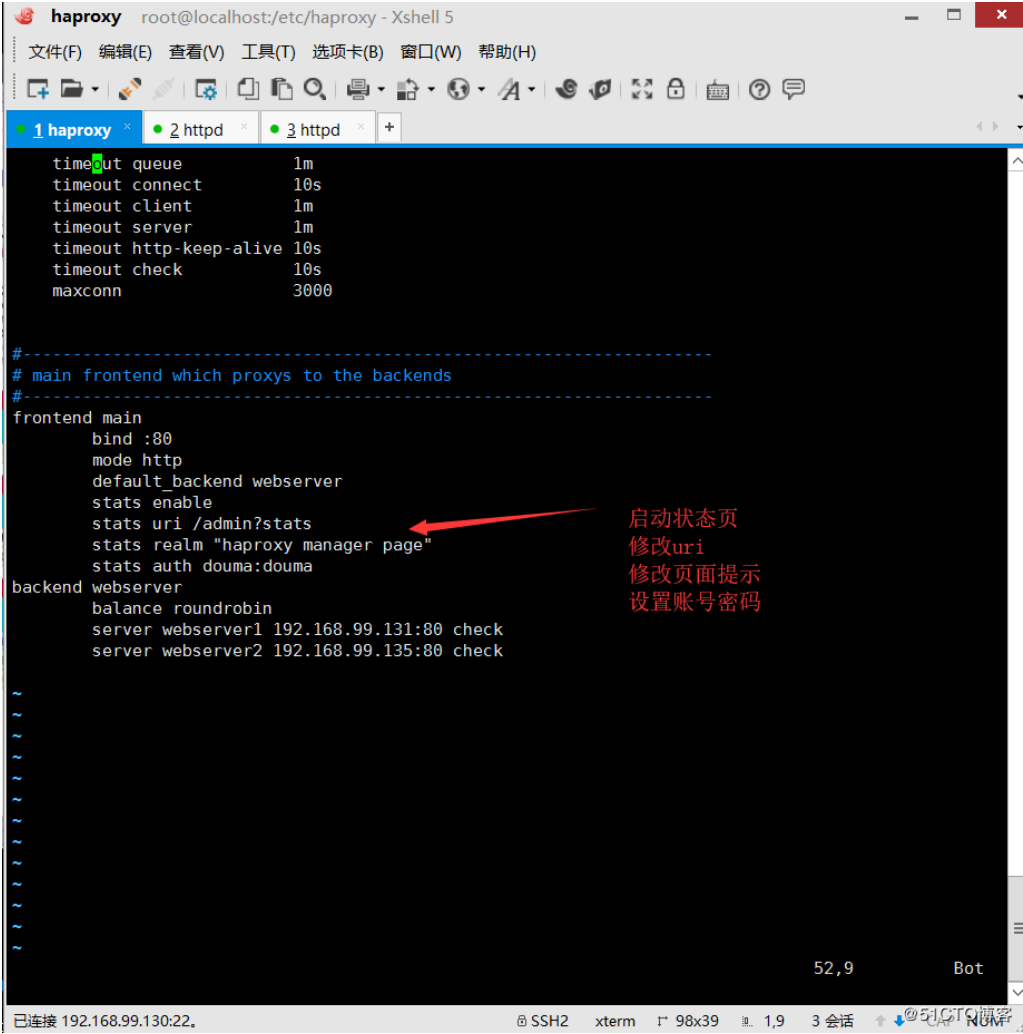
启用统计页；基于默认的参数启用stats page；

- ```
- stats uri      : /haproxy?stats  访问页默认地址，支持重写
- stats realm   : "HAProxy Statistics"
- stats auth    : no authentication
- stats scope   : no restriction
```

在线客服



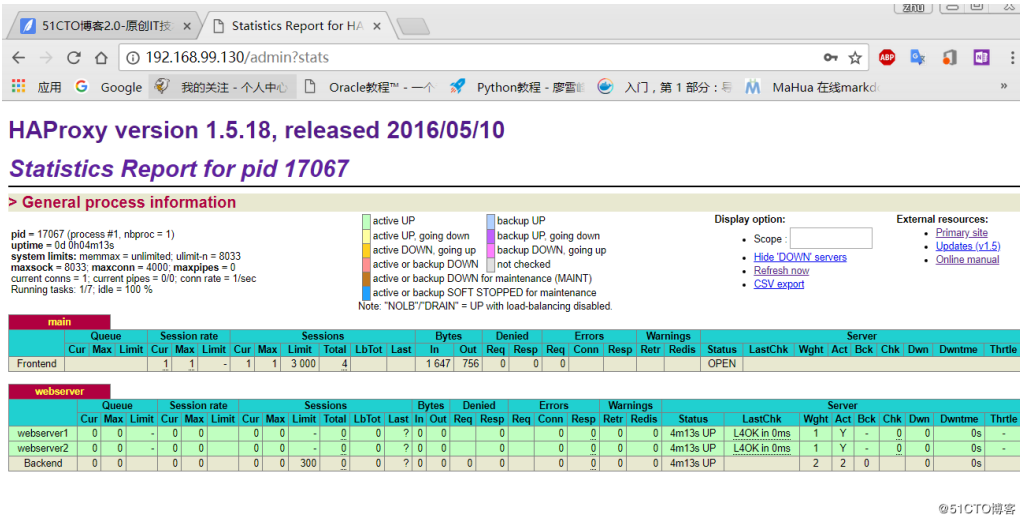
示例



@51CTO博客



在线  
客服



**maxconn** <conns>: 为指定的frontend定义其最大并发连接数；默认为2000；  
Fix the maximum number of concurrent connections on a frontend.

**mode** { tcp|http|health }:定义haproxy的工作模式；  
tcp: 基于layer4实现代理；可代理mysql, postgresql, ssh, ssl等协议；  
http: 仅当代理的协议为http时使用；  
health: 工作为健康状态检查的响应模式，当连接请求到达时回应“OK”后即断开连接；  
例子：

```
listen ssh
    bind :22011
    balance leastconn
    mode tcp
    server sshsrv1 192.168.99.131:22 check
    server sshsrv2 192.168.99.135:22 check
```

**forwardfor** [ except <network> ] [ header <name> ] [ if-none ]  
在由haproxy发往后端主机的请求报文中添加“X-Forwarded-For”首部，其值前端客户端的地址；用于向后端主发送真实的客户端IP；  
[ except <network> ]: 请求报来自此处指定的网络时不予添加此首部；  
[ header <name> ]: 使用自定义的首部名称，而非“X-Forwarded-For”  
示例：



在线  
客服

haproxy配置文件

1 haproxy2 httpd3 httpd+

```
chroot      /var/lib/haproxy
pidfile     /var/run/haproxy.pid
maxconn     4000
user        haproxy
group       haproxy
daemon

# turn on stats unix socket
stats socket /var/lib/haproxy/stats

#-----
# common defaults that all the 'listen' and 'backend' sections will
# use if not designated in their block
#-----
defaults
    mode                http
    log                 global
    option              httplog
    option              dontlognull
    option http-server-close
    option forwardfor   except 127.0.0.0/8
    option               redispatch
    retries              3
    timeout http-request 10s
    timeout queue        1m
    timeout connect      10s
    timeout client        1m
    timeout server       1m
    timeout http-keep-alive 10s
    timeout check         10s
    maxconn             3000

#-----
# main frontend which proxys to the backends
#-----
frontend main
```

64,176%  
@51CTO博客

系统默认就进行了设定，这里排除了127.0.0.0网段的ip

后端httpd配置文件（修改后要重启httpd服务）

1 haproxy2 httpd3 httpd+

```
# container, error messages relating to that virtual host will be
# logged here.  If you *do* define an error logfile for a <VirtualHost>
# container, that host's errors will be logged there and not here.
#
ErrorLog "logs/error_log"

#
# LogLevel: Control the number of messages logged to the error_log.
# Possible values include: debug, info, notice, warn, error, crit,
# alert, emerg.
#
LogLevel warn

<IfModule log_config_module>
    #
    # The following directives define some format nicknames for use with
    # a CustomLog directive (see below).
    #
    LogFormat "%(X-Forwarded-For)i %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined
    #
    LogFormat "%h %l %u %t \"%r\" %>s %b" common

    <IfModule logio_module>
        # You need to enable mod_logio.c to use %I and %O
        LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\" %I %O" combinedio
    </IfModule>

    #
    # The location and format of the access logfile (Common Logfile Format).
    # If you do not define any access logfiles within a <VirtualHost>
    # container, they will be logged here.  Contrariwise, if you *do*
    # define per-<VirtualHost> access logfiles, transactions will be
    # logged therein and *not* in this file.
    #
    #CustomLog "logs/access_log" common

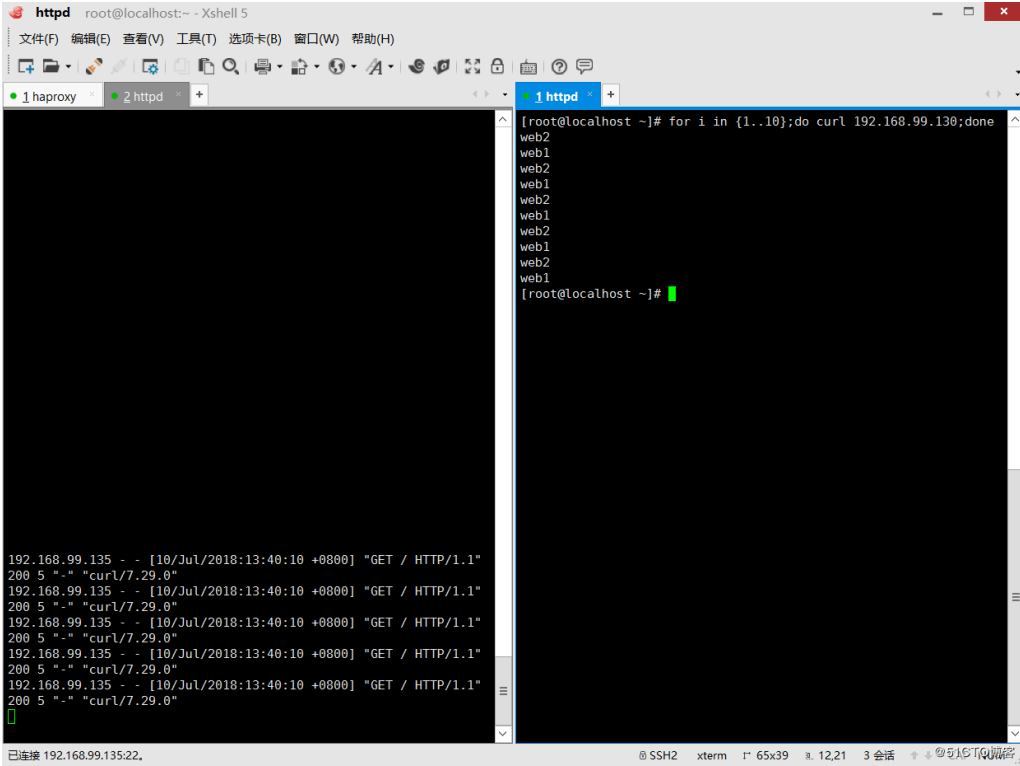
    #
    # If you prefer a logfile with access, agent, and referer information
    #/etc/httpd/conf/httpd.conf" 353L, 11770C
```

196,556%  
@51CTO博客

送图书

在线客服

测试效果：记录了原地址。

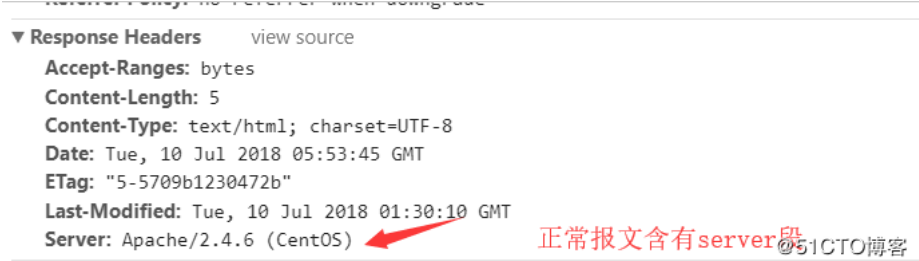


**rspadd** <string> [(if | unless) <cond>]在http响应首部添加字段信息  
rspadd X-Via:\ douma

**reqdel** <search> [(if | unless) <cond>]在http响应包首都中删除匹配到的信息  
rspidel (i忽略字符大小写) Server.\*

示例：可以用来实现删除server信息，以防暴露站点漏洞

修改前



haproxy配置文件



在线客服



```
#-----
defaults
    mode                http
    log                 global
    option              httplog
    option              dontlognull
    option http-server-close
    option forwardfor   except 127.0.0.0/8
    option              redispatch
    retries             3
    timeout http-request 10s
    timeout queue       1m
    timeout connect     10s
    timeout client      1m
    timeout server      1m
    timeout http-keep-alive 10s
    timeout check       10s
    maxconn             3000

#-----
# main frontend which proxys to the backends
#-----
frontend main
    bind :80
    mode http
    default_backend webserver
    rspadd X-via:douma
    rspidel Server.*
backend webserver
    balance roundrobin
    server webserver1 192.168.99.131:80 check
    server webserver2 192.168.99.135:80 check
```

添加一个x-via段  
删除server段

69,1 @51CTO博客

新效果：

× Headers Preview Response Cookies Timing

headers only no refresh when using ajax

▼ Response Headers view source

Accept-Ranges: bytes

Content-Length: 5

Content-Type: text/html; charset=UTF-8

Date: Tue, 10 Jul 2018 05:56:01 GMT

Etag: "5-5709b1230472b"

Last-Modified: Tue, 10 Jul 2018 01:30:10 GMT

X-via: douma

server段被删除了，多了一个x-via段

@51CTO博客



## ACL访问控制

### 指令结构

acl <aclname> <criterion> [flags] [operator] [<value>] ...

aclname：设定acl的名称

value：设定acl的值

- boolean：布尔型
- integer or integer range：整数或者范围
- ip address /network：ip地址或者网络地址
- string：字符串
  - exact：精确匹配
  - substring：子串匹配
  - suffix：后缀匹配
  - prefix：前缀匹配
  - subdir：子路径匹配
  - domain：子域匹配
- regular expression：正则表示模式匹配
- hex block：十六进匹配

flag标志位

<flags>  
-i：在子串匹配时候忽略大小写

在线  
客服

-u : 要求ACL使用唯一名称

-- : force end of flags. Useful when a string looks like one of the flags

[operator]

匹配整数值: eq、ge、gt、le、lt

匹配字符串:

- exact match (-m str) : the extracted string must exactly match
- substring match (-m sub) : the patterns are looked up inside the string
- prefix match (-m beg) : the patterns are compared with the beginning
- suffix match (-m end) : the patterns are compared with the end
- subdir match (-m dir) : the patterns are looked up inside the directory
- domain match (-m dom) : the patterns are looked up inside the domain

acl作为条件时的逻辑关系:

- AND (implicit)
- OR (explicit with the "or" keyword or the "||" operator)
- Negation with the exclamation mark ("!")

if invalid\_src invalid\_port 要求同时满足

if invalid\_src || invalid\_port 或表示

if ! invalid\_src invalid\_port 取反, 不满足第一个但是满足第二个

<criterion> :

dst : ip 目标ip或者范围

dst\_port : integer 目标端口或范围

src : ip 源ip或者范围

src\_port : integer 源端口后缀范围

acl invalid\_src src 172.16.200.2

path : string 七层检查  
/path;<params>

- path : exact string match 精确匹配
- path\_beg : prefix match 前缀匹配
- path\_dir : subdir match 子串匹配
- path\_dom : domain match 子域匹配
- path\_end : suffix match 路径后缀匹配
- path\_len : length match 长度匹配
- path\_reg : regex match 正则匹配
- path\_sub : substring match 子串匹配



示例

```
path_beg /images/ 除去ip:port之后以/images/开头
path_end .jpg .jpeg .png .gif url以jpg jpeg png gif后缀的文件
path_reg ^/images.*\.jpeg$ images开头 jpeg结尾
path_sub image
path_dir jpegs
path_dom ilinux
```

/images/jpegs/20180312/logo.jpg

url : string 对url进行匹配

- url : exact string match
- url\_beg : prefix match
- url\_dir : subdir match
- url\_dom : domain match
- url\_end : suffix match
- url\_len : length match
- url\_reg : regex match
- url\_sub : substring match

在线客服

req.hdr([<name>[,<occ>]]) : string

hdr([<name>[,<occ>]]) : exact string match



```
HAProxy入门及常用配置模拟测试-一条逐渐变咸的咸鱼-51CTO博客
hdr_end([<name>[,<occ>]]) : suffix match
hdr_len([<name>[,<occ>]]) : length match
hdr_reg([<name>[,<occ>]]) : regex match
hdr_sub([<name>[,<occ>]]) : substring match
```

©著作权归作者所有：来自51CTO博客作者Doumadouble的原创作品，如需转载，请注明出处，否则将追究法律责任

haproxy      负载均衡      基础知识      应用服务

4      收藏      分享

上一篇：NGINX从入门到放弃（原理与编...      下一篇：keepalived高可用hap...



Doumadouble  
38篇文章，28W+人气，0粉丝  
一入运维深似海，从此服务为伴侣



提问和评论都可以，用心的回复会被更多人看到和认可

Ctrl+Enter 发布      取消      发布

1条评论

按时间正序 | 按时间倒序



心无倦时  
1楼 2018-07-11 08:38:16  
大佬很快,这么早就能出博客了

推荐专栏

更多



基于Python的DevOps实战  
自动化运维开发新概念  
共20章 | 抚琴煮酒  
¥51.00      355人订阅

订 阅



全局视角看大型园区网  
路由交换+安全+无线+优化+运维  
共40章 | 51CTO夏杰  
¥51.00      1015人订阅

订 阅



网工2.0晋级攻略 —— 零基础入门Python/A...  
网络工程师2.0进阶指南  
共30章 | 姜汁啤酒

订 阅





负载均衡高手炼成记

高并发架构之路

共15章 | sery

¥51.00    446人订阅

订 阅



带你玩转高可用

前百度高级工程师的架构高可用实战

共15章 | 曹林华

¥51.00    423人订阅

订 阅

猜你喜欢

tomcat的session会话保持方案

通过RKE 安装kubernetes

监控之路5-zabbix定义一次完整的监控

Linux 四剑客介绍和案例

keepalived高可用haproxy+varnish+lnmp实现站点搭建...

Linux Redis 高可用之主从复制

使用elasticdump迁移数据到新es集群

kubernetes1.13.1集群集成harbor-helm



在线  
客服