# Haproxy 1.8.9 安装配置手册

## 安装包准备

安装包下载

进入到<https://www.haproxy.org/> 下载1.8.9版本

将下载好的安装包上传到

/opt/aspire/product/dicmp/haproxy目录

执行解压命令

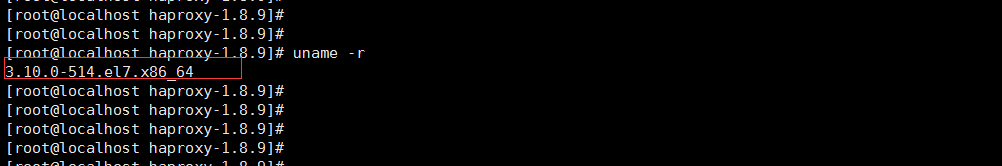
tar -zxvf haproxy-1.8.9.tar.gz

进入到/opt/aspire/product/dicmp/haproxy/haproxy-1.8.9目录

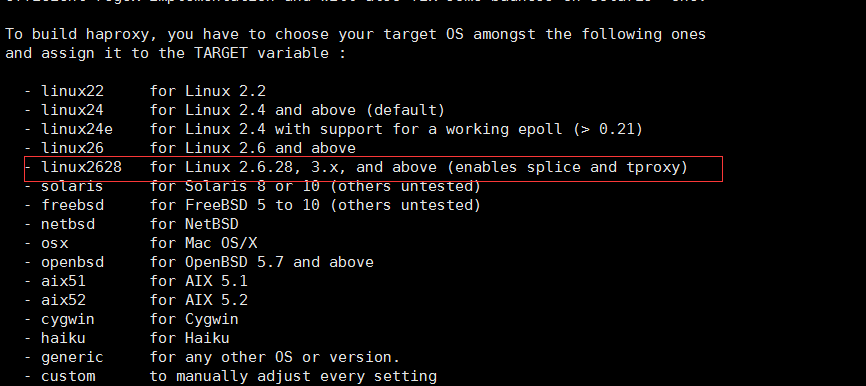
## 安装

查看系统的内核版本

执行命令 uname -r



查看README文件复制安装命令



通过README文件可以查找到内核版本对应的make参数

make TARGET=linux2628 USE\_PCRE=1 USE\_OPENSSL=1 USE\_ZLIB=1 PREFIX=/usr/local/haproxy

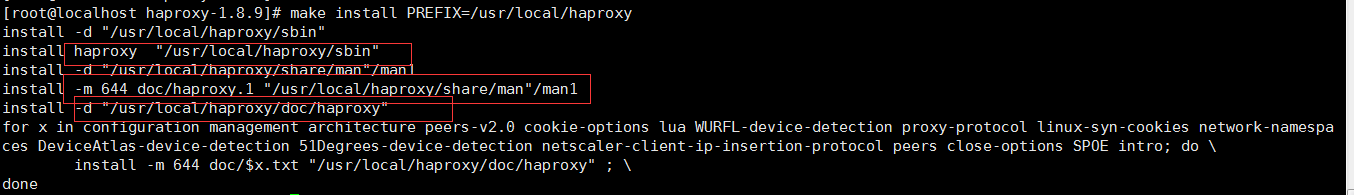
如果make没有报错，继续执行make install命令：

切换到root权限执行

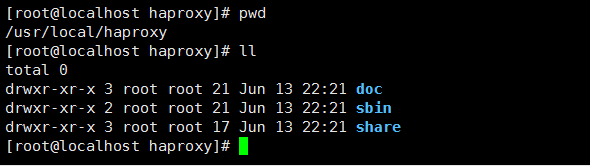
make install PREFIX=/usr/local/haproxy

或者直接执行

sudo make install PREFIX=/usr/local/haproxy

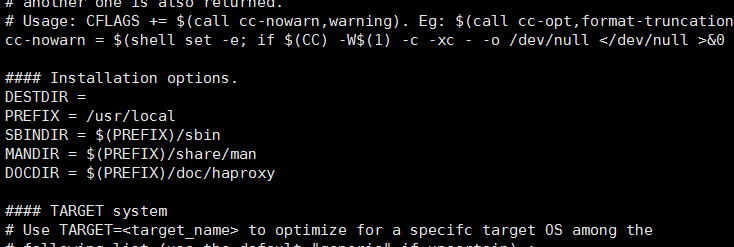


查看make install输出，我们可以知道，make install会在/usr/local/haproxy中创建sbin，share，man三个目录，进入到/usr/local/haproxy检查结果：



说明我们已经安装完成了。

Makefile默认的安装目录是/usr/local，未指定PREFIX参数的请检查默认安装目录。



## 配置

由于新版中，haproxy不会在etc下面生产默认的配置文件了，在这里我们就需要自己创建一个配置文件

### 参考配置

global

log 127.0.0.1 local0 info

daemon

user haproxy

group haproxy

global

log 127.0.0.1 local0 info

daemon

user haproxy

group haproxy

maxconn 4096

defaults

mode http

timeout connect 5000ms

timeout client 50000ms

timeout server 50000ms

option redispatch

option abortonclose

option dontlognull

listen status

bind 0.0.0.0:8080

mode http

log global

stats enable

stats refresh 30s

stats uri /admin\_status

stats realm Private lands

stats auth admin:admin

stats admin if TRUE

stats hide-version

frontend http-in

mode http

bind \*:80

default\_backend servers

backend servers

server server1 127.0.0.1:8090 check maxconn 32

server server2 127.0.0.1:8090 check maxconn 32

server server3 127.0.0.1:8090 check maxconn 32

### 配置文件结构说明

haproxy的配置文件可划分为5段：

global 全局配置

default 默认配置

listen 同时拥有frontend与backend的功能

frontend 代理前端

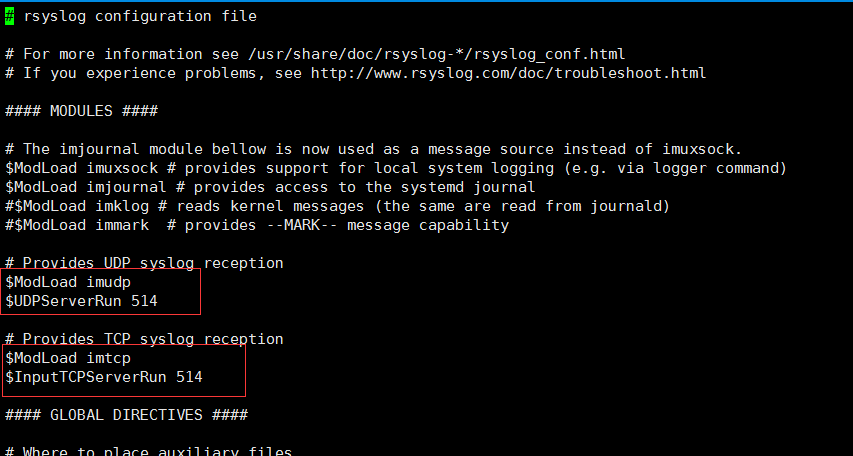
backend 后端真实服务器组

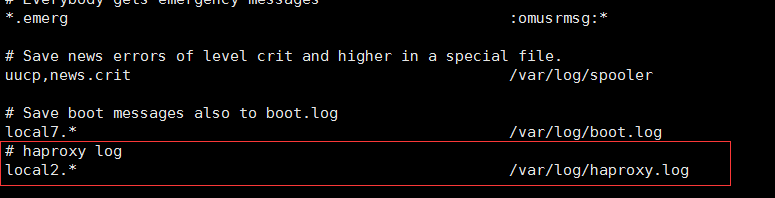
## 打开系统日志

### 修改配置

vi /etc/rsyslog.conf

做如下修改





### 重启rsyslog

systemctl restart rsyslog

## 启动

执行以下命令启动haproxy：

/usr/local/haproxy/sbin/haproxy -D -f /usr/local/haproxy/haproxy.cfg

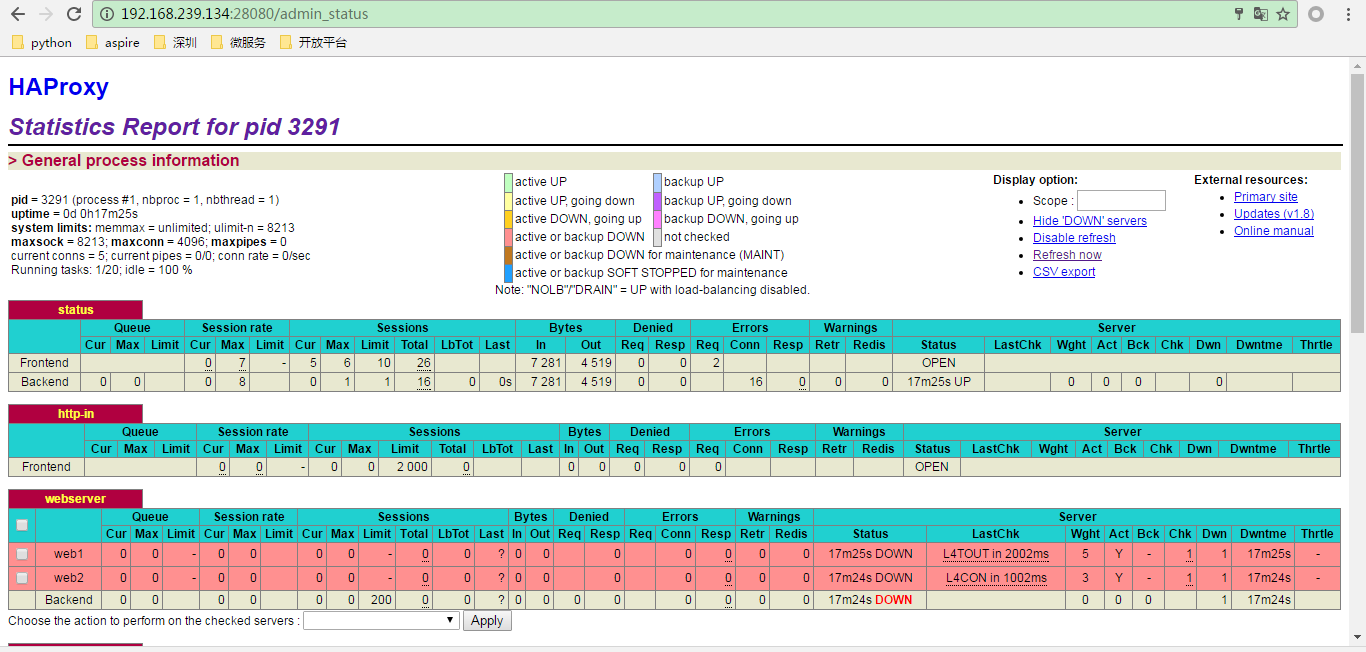
## 运行状态检查

方法一：

在浏览器中输入

<http://192.168.239.134:28080/admin_status>

输入帐号密码后可以看到如下页面:



方法二：

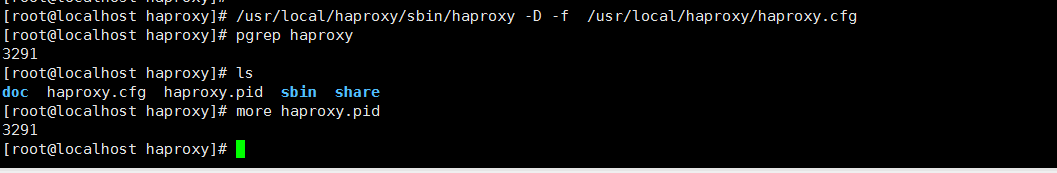
执行启动命令如果没有报错说明haproxy已经启动了，可以通过

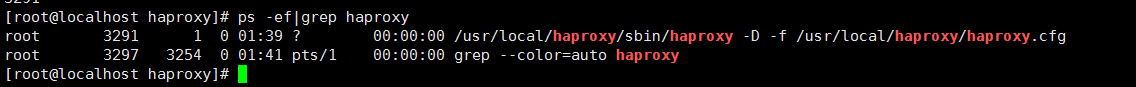
pgrep haproxy

查询到haproxy的进程pid，也可以查看/usr/local/haproxy/haproxy.pid文件

cat /usr/local/haproxy/haproxy.pid

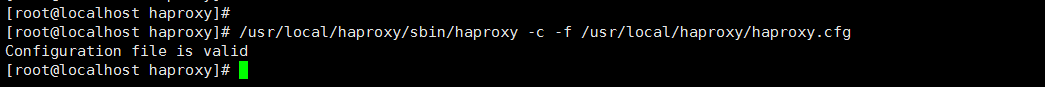
当然也可以通过ps –ef|grep haproxy查看咯





## 配置检查

/usr/local/haproxy/sbin/haproxy -c -f /usr/local/haproxy/haproxy.conf



## 重新加载进程

等待旧的连接处理完成后再启动新的进行

/usr/local/haprpxy/sbin/haproxy -D -f /usr/local/haprpxy/conf/haproxy.cfg -sf `pgrep haproxy`

不等待旧进程处理完成，直接启动新进程

/usr/local/haprpxy/sbin/haproxy -D -f /usr/local/haprpxy/conf/haproxy.cfg -st `pgrep haproxy`

## 停止

pkill haproxy

平滑停止

kill -s SIGUSR1  `pgrep haproxy`

立即停止 不等进程处理完当前请求

kill -s SIGTERM   `pgrep haproxy`

强制杀死进程

kill -9 $pid

## 负载均衡说明

一、roundrobin，表示简单的轮询，每个服务器根据权重轮流使用，在服务器的处理时间平均分配的情况下这是最流畅和公平的算法。该算法是动态的，对于实例启动慢的服务器权重会在运行中调整。

二、static-rr，表示根据权重，建议关注；每个服务器根据权重轮流使用，类似roundrobin，但它是静态的，意味着运行时修改权限是无效的。另外，它对服务器的数量没有限制。

三、leastconn，表示最少连接者先处理，建议关注；leastconn建议用于长会话服务，例如LDAP、SQL、TSE等，而不适合短会话协议。如HTTP.该算法是动态的，对于实例启动慢的服务器权重会在运行中调整。

四、source，表示根据请求源IP，建议关注；对请求源IP地址进行哈希，用可用服务器的权重总数除以哈希值，根据结果进行分配。

只要服务器正常，同一个客户端IP地址总是访问同一个服务器。如果哈希的结果随可用服务器数量而变化，那么客户端会定向到不同的服务器；

该算法一般用于不能插入cookie的Tcp模式。它还可以用于广域网上为拒绝使用会话cookie的客户端提供最有效的粘连；

该算法默认是静态的，所以运行时修改服务器的权重是无效的，但是算法会根据“hash-type”的变化做调整。

五、uri，表示根据请求的URI；表示根据请求的URI左端（问号之前）进行哈希，用可用服务器的权重总数除以哈希值，根据结果进行分配。

只要服务器正常，同一个URI地址总是访问同一个服务器。

一般用于代理缓存和反病毒代理，以最大限度的提高缓存的命中率。该算法只能用于HTTP后端；

该算法一般用于后端是缓存服务器；

该算法默认是静态的，所以运行时修改服务器的权重是无效的，但是算法会根据“hash-type”的变化做调整。

六、url\_param，表示根据请求的URl参数'balance url\_param' requires an URL parameter name

在HTTP GET请求的查询串中查找<param>中指定的URL参数，基本上可以锁定使用特制的URL到特定的负载均衡器节点的要求；

该算法一般用于将同一个用户的信息发送到同一个后端服务器；

该算法默认是静态的，所以运行时修改服务器的权重是无效的，但是算法会根据“hash-type”的变化做调整。

七、hdr(name)，表示根据HTTP请求头来锁定每一次HTTP请求；

在每个HTTP请求中查找HTTP头<name>，HTTP头<name>将被看作在每个HTTP请求，并针对特定的节点；

如果缺少头或者头没有任何值，则用roundrobin代替；

该算法默认是静态的，所以运行时修改服务器的权重是无效的，但是算法会根据“hash-type”的变化做调整。

八、rdp-cookie(name)，表示根据据cookie(name)来锁定并哈希每一次TCP请求。

为每个进来的TCP请求查询并哈希RDP cookie<name>；

该机制用于退化的持久模式，可以使同一个用户或者同一个会话ID总是发送给同一台服务器。

如果没有cookie，则使用roundrobin算法代替；

该算法默认是静态的，所以运行时修改服务器的权重是无效的，但是算法会根据“hash-type”的变化做调整。

#其实这些算法各有各的用法，我们平时应用得比较多的应该是roundrobin、source和lestconn。

haproxy负载均衡算法

## acl规则定义说明

########ACL策略定义#########################

1、#如果请求的域名满足正则表达式返回true -i是忽略大小写

acl denali\_policy hdr\_reg(host) -i ^(www.inbank.com|image.inbank.com)$

2、#如果请求域名满足www.inbank.com 返回 true -i是忽略大小写

acl tm\_policy hdr\_dom(host) -i www.inbank.com

3、#在请求url中包含sip\_apiname=，则此控制策略返回true,否则为false

acl invalid\_req url\_sub -i sip\_apiname=#定义一个名为invalid\_req的策略

4、#在请求url中存在timetask作为部分地址路径，则此控制策略返回true,否则返回false

acl timetask\_req url\_dir -i timetask

5、#当请求的header中Content-length等于0时返回 true

acl missing\_cl hdr\_cnt(Content-length) eq 0

#########acl策略匹配相应###################

1、#当请求中header中Content-length等于0 阻止请求返回403

block if missing\_cl

2、#block表示阻止请求，返回403错误，当前表示如果不满足策略invalid\_req，或者满足策略timetask\_req，则阻止请求。

block if !invalid\_req || timetask\_req

3、#当满足denali\_policy的策略时使用denali\_server的backend

use\_backend denali\_server if denali\_policy

4、#当满足tm\_policy的策略时使用tm\_server的backend

use\_backend tm\_server if tm\_policy

5、#reqisetbe关键字定义，根据定义的关键字选择backend

reqisetbe ^Host:\ img dynamic

reqisetbe ^[^\ ]\*\ /(img|css)/ dynamic

reqisetbe ^[^\ ]\*\ /admin/stats stats

6、#以上都不满足的时候使用默认mms\_server的backend

default\_backend mms

haproxy acl定义