ACL 匹配方式列表：

path - 完全匹配 URL 路径，不包含参数部分（?后面的内容）
 #下面是简写，也可通过 -m xxx 替代，-m xxx 是通用方法
 path_beg - 匹配 URL 路径的开头部分 (匹配从 path 开头部分开始连续的字符串)
 path_end - 匹配 URL 路径的结束部分 (匹配从 path 任意字符到 path 结尾连续的字符串)
 path_dir - 如果 URL 路径包含指定的目录名，则匹配成功 （匹配以/分割的所有字符串，包括最后一段包含文件名的部分）
 path_reg - 使用正则表达式匹配 URL 路径
  -- 使用示例：acl url_static path_reg ^.*/test/+[^/]+$ -- 匹配最后一个目录是 test 的 path
  -- 使用示例：acl url_static path_reg ^.*/test/+[^/]*$ -- 匹配最后一个目录是 test，且也包含/test/即是路径末尾的 path
 path_sub - 匹配 URL 路径中包含的子字符串 （匹配 path 中包含的任意字符）
  -- 使用示例：acl url_static path_sub logo. -- 匹配文件名为 logo 的 path
 path_dom - 如果 URL 路径包含指定的字符串，则匹配成功 （匹配以 /|.|?|#|: 分割的所有字符串，且允许字符串包含分隔符）
    #注意，这里的包含指的是只要指定字符串的边界是以上分隔符中的，则符合条件
 path_len - 匹配 URL 路径部分的长度（字符数量,包括/分隔符）

url - 对整个 URL 进行匹配，包括?后面的请求参数部分
 url_xxx xxx 支持的参数和 path 完全相同

hdr(xxx) - 匹配 HTTP 请求头的值
 hdr_beg(xxx) - 匹配 HTTP 请求头的开始部分
 hdr_end(xxx) - 匹配 HTTP 请求头的结束部分
 hdr_sub(xxx) - 匹配 HTTP 请求头包含的子字符串
 hdr_dom(xxx) - 匹配请求头中的域名部分

 hdr_beg(host) - 匹配从 host 开头部分开始连续的字符串
 hdr_end(host) - 匹配从 host 任意字符到 host 结尾连续的字符串
 hdr_dom(host) - 匹配 host 以.|: 分割的所有字符串

cookie(xxx) - 根据 Cookie 的内容进行匹配

urlp(xxx) -匹配请求参数

src - 匹配客户端的源 IP 地址
dst - 匹配目标服务器的 IP 地址
dst_port - 目标 PORT
src_port - 源 PORT
method - 匹配 HTTP 请求方法，如 GET、POST 等

ssl_fc - 检查是否为 SSL 连接（返回值为 0 或 1）。
query - 匹配 URL 中的查询字符串（url 中?后面的全部内容部分）

ACL 配置语法

访问控制列表（ACL，Access Control Lists）是一种基于包过滤的访问控制技术，它可以根据设定的条件对经过服 务器传输的数据包进行过滤(条件匹配)，即对接收到的报文进行匹配和过滤，基于请求报文头部中的源地址、源端 口、目标地址、目标端口、请求方法、URL、文件后缀等信息内容进行匹配并执行进一步操作，比如允许其通过或 丢弃。 http://cbonte.github.io/haproxy-dconv/2.0/configuration.html#7

1、ACL 配置选项

acl <aclname> <criterion> [flags] [operator] [<value>]
acl 名称 匹配规范 匹配模式 具体操作符 操作对象类型

匹配内容：请求头、请求方式、请求 URL、请求参数、请求内容、source ip、source port、dest ip、dest port

1.1 ACL-Name

acl image_service hdr_dom(host) -i img.magedu.com
ACL 名称，可以使用大字母 A-Z、小写字母 a-z、数字 0-9、冒号：、点.、中横线和下划线，并且严格区分大小写，必须 Image_site 和 image_site 完全是两个 acl。

1.2 ACL-criterion

定义 ACL 匹配规范
hdr（[<name> [，<occ>]]）：完全匹配字符串,header 的指定信息
hdr_beg（[<name> [，<occ>]]）：前缀匹配，header 中指定匹配内容的 begin
hdr_end（[<name> [，<occ>]]）：后缀匹配，header 中指定匹配内容 end
hdr_dom（[<name> [，<occ>]]）：域匹配，header 中的 domain name
hdr_dir（[<name> [，<occ>]]）：路径匹配，header 的 uri 路径
hdr_len（[<name> [，<occ>]]）：长度匹配，header 的长度匹配
hdr_reg（[<name> [，<occ>]]）：正则表达式匹配，自定义表达式(regex)模糊匹配
hdr_sub（[<name> [，<occ>]]）：子串匹配，header 中的 uri 模糊匹配
dst 目标 IP
dst_port 目标 PORT
src 源 IP
src_port 源 PORT

示例：
hdr(<string>) 用于测试请求头部首部指定内容
hdr_dom(host) 请求的 host 名称，如 www.magedu.com
hdr_beg(host) 请求的 host 开头，如 www. img. video. download. ftp.
hdr_end(host) 请求的 host 结尾，如 .com .net .cn
path_beg 请求的 URL 开头，如/static、/images、/img、/css
path_end 请求的 URL 中资源的结尾，如 .gif .png .css .js .jpg .jpeg

有些功能是类似的，比如以下几个都是匹配用户请求报文中 host 的开头是不是 www：
acl short_form hdr_beg(host) www.
acl alternate1 hdr_beg(host) -m beg www.
acl alternate2 hdr_dom(host) -m beg www.
acl alternate3 hdr(host) -m beg www.

1.3 ACL-flags

ACL 匹配模式
-i 不区分大小写
-m 使用指定的 pattern 匹配方法
类型描述
found 只检查流中是否存在请求的样本。例如，使用这个来检查 URL 参数是否存在，而不关心它的值
bool
将样本匹配为布尔值。此方法仅适用于返回布尔值或整数值的获取。值 0 或 false 不匹配，所有其他值匹配。
int 将样本匹配为整数。它可以应用于整数和布尔样本。布尔值 false 为整数 0，true 为整数 1。
ip 将样本匹配为 IPv4 或 IPv6 地址。只兼容 IP 地址样例。
bin 将样本与表示二进制序列的十六进制字符串进行匹配。它可以应用于二进制或字符串样本。
len 将样本的长度（字符数）匹配为整数。它可以应用于二进制或字符串样本。
str 精确的字符串匹配。它可以应用于二进制或字符串样本。
sub 子字符串匹配：检查样例是否包含至少一个所提供的字符串模式。它可以应用于二进制或字符串样本。
reg 根据正则表达式列表匹配样本。这可以用于二进制或字符串样本。
beg 前缀匹配：检查样本的开头是否与所提供的任何模式相似。它可以应用于二进制或字符串样本。
end 后缀匹配：检查样例是否与所提供的任何模式相同。它可以应用于二进制或字符串样本。
dir 子目录匹配：检查样本中斜杠分隔的部分是否与所提供的模式之一完全匹配。它可以应用于二进制或字符串样本。
dom 域匹配：检查样例中以 .|/|? 分隔的部分是否与所提供的模式之一完全匹配。它可以应用于二进制或字符串样本。
-n 不做 DNS 解析, 和-f 一起使用，只允许文件中配置 ip 地址，不允许配置域名
-u 禁止 acl 重名，否则多个同名 ACL 匹配或关系
-f 从指定的文件中加载模式 #匹配的值存储于指定文件中
-- 标志符的强制结束标记，在模式中的字符串像标记符时使用；比如：acl has_dash_i url_sub -- -i

1.4 ACL-operator

ACL 操作符
整数比较：eq、ge、gt、le、lt
字符比较：

- exact match (-m str) :字符串必须完全匹配模式
- substring match (-m sub) :在提取的字符串中查找模式，如果其中任何一个被发现，ACL 将匹配
- prefix match (-m beg) :在提取的字符串首部中查找模式，如果其中任何一个被发现，ACL 将匹配 (匹配从目标字符串开头部分开始连续的字符串)
- suffix match (-m end) :将模式与提取字符串的尾部进行比较，如果其中任何一个匹配，则 ACL 进行匹配 (匹配从目标字符串任意字符到 path 结尾连续的字符串)
- subdir match (-m dir) :查看提取出来的用斜线分隔（“/”）的字符串，如果其中任何一个匹配，则 ACL 进行匹配
- domain match (-m dom) :查找提取的用（“.|/|?”）分隔的字符串，如果其中任何一个匹配，则 ACL 进行匹配

二进制比较：

-m bin 48656c6c6f0a

acl hello payload(0,6) -m bin 48656c6c6f0a #必须用一系列十六进制数字传递，这些数字必须是偶数个，每对数字代表一个字节

1.5 ACL-value

value 的类型

The ACL engine can match these types against patterns of the following types :

- Boolean #布尔值
- integer or integer range #整数或整数范围，比如用于匹配端口范围
- IP address / network #IP 地址或 IP 范围, 192.168.0.1 ,192.168.0.1/24
- string--> www.magedu.com
  exact –精确比较
  substring—子串
  suffix-后缀比较
  prefix-前缀比较
  subdir-路径， /wp-includes/js/jquery/jquery.js
  domain-域名，www.magedu.com
- regular expression #正则表达式
- hex block #16 进制

2、ACL 调用方式

ACL 调用方式：

- 与：隐式（默认）使用
- 或：使用“or” 或 “||”表示
- 否定：使用“!“ 表示

示例：

if valid_src valid_port #与关系,A 和 B 都要满足为 true

if invalid_src || invalid_port #或，A 或者 B 满足一个为 true

if ! invalid_src #非，取反，A 和 B 哪个也不满足为 true

3、ACL 的实例

3.1 基于 host 匹配

hdr(host)包含端口号，hdr_dom(host)不包含端口号

3.1 .1 匹配 www.magedu.net 域名的请求

```
listen web_host
 bind 192.168.32.204:80
 mode http
 balance roundrobin
 log global
 option httplog
 acl web_host hdr_dom(host) www.magedu.net
 use_backend magedu_host if web_host
 default_backend default_web

backend magedu_host
 mode http
 server web1 10.0.0.201:80 check inter 2000 fall 3 rise 5

backend default_web
 mode http
 server web1 10.0.0.202:80 check inter 2000 fall 3 rise 5
```

3.1 .2 匹配 113.44.228.35 的请求

```
listen web_host
 bind 192.168.32.204:80
 mode http
 balance roundrobin
 log global
 option httplog
 acl web_host hdr_dom(host) 113.44.228.35
 use_backend magedu_host if web_host
 default_backend default_web

backend magedu_host
 mode http
 server web1 10.0.0.201:80 check inter 2000 fall 3 rise 5

backend default_web
 mode http
 server web1 10.0.0.202:80 check inter 2000 fall 3 rise 5
```

3.1 .3 匹配以 www 开头的请求

```
listen web_host
```

```
 bind 192.168.32.204:80
 mode http
 balance roundrobin
 log global
 option httplog
 acl web_host  hdr_dom(host) -m beg www.
 use_backend magedu_host if web_host
 default_backend default_web

backend magedu_host
 mode http
 server web1 10.0.0.201:80 check inter 2000 fall 3 rise 5

backend default_web
 mode http
 server web1 10.0.0.202:80 check inter 2000 fall 3 rise 5
```

3.2 基于源 IP 或子网调度访问

```
将指定的源地址调度至指定的 web 服务器组。
frontend  main *:5000
 mode http
 balance roundrobin
 log global
 option httplog
 acl ip_range_test src 10.0.0.0/16 192.168.32.201
 use_backend magedu_host if ip_range_test
 default_backend default_web

backend magedu_host
mode http
server web1 10.0.0.201 check inter 2000 fall 3 rise 5

backend default_web
mode http
server web1 10.0.0.202:80 check inter 2000 fall 3 rise 5
```

3.3 基于源地址的访问控制

```
拒绝指定 IP 或者 IP 范围访问
frontend  main *:5000
 mode http
 balance roundrobin
 log global
```

```
 option httplog
 acl block_test src 192.168.32.203 192.168.0.0/24
 block if block_test
 default_backend default_web

backend magedu_host
 mode http
 server web1 10.0.0.201 check inter 2000 fall 3 rise 5

backend default_web
 mode http
 server web1 10.0.0.202:80 check inter 2000 fall 3 rise 5
```

3.4 基于目标端口号

```
拒绝指定 IP 或者 IP 范围访问
frontend  main *:5000
 mode http
 balance roundrobin
 log global
 option httplog
 acl url_static dst_port 5000
 use_backend static  if url_static
 default_backend default_web

backend magedu_host
 mode http
 server web1 10.0.0.201 check inter 2000 fall 3 rise 5

backend default_web
 mode http
 server web1 10.0.0.202:80 check inter 2000 fall 3 rise 5
```

3.5 匹配浏览器类型

```
匹配客户端浏览器，将不同类型的浏览器调动至不同的服务器组
frontend  main *:5000
 mode http
 balance roundrobin
 log global
 option httplog
 acl web_host hdr_dom(host) www.magedu.net
 use_backend magedu_host if web_host
 acl redirect_test hdr(User-Agent) -m sub -i "Mozilla/5.0 (Windows NT 6.1; WOW64;Trident/7.0;
rv:11.0) like Gecko"
```

```
 redirect prefix http://192.168.32.201 if redirect_test    #带后缀跳转
 #redirect location http://192.168.32.201 if redirect_test  #直接跳转不带后缀
 default_backend default_web

backend magedu_host
 mode http
 server web1 10.0.0.201 check inter 2000 fall 3 rise 5

backend default_web
 mode http
 server web1 10.0.0.202:80 check inter 2000 fall 3 rise 5
```

3.6 基于文件后缀名实现动静分离

```
frontend  main *:5000
 mode http
 balance roundrobin
 log global
 option httplog
 acl php_server path_end -i .php
 use_backend php_server_host if php_server
 acl image_server path_end -i .jpg .png .jpeg .gif
 use_backend image_server_host if image_server
 default_backend default_web

backend php_server_host
 mode http
 server web1 10.0.0.203 check inter 2000 fall 3 rise 5

backend image_server_host
 mode http
 server web1 10.0.0.201 check inter 2000 fall 3 rise 5

backend default_web
 mode http
 server web1 10.0.0.202:80 check inter 2000 fall 3 rise 5
```

3.7 匹配访问路径实现动静分离

```
frontend  main *:5000
 mode http
 balance roundrobin
 log global
 option httplog
```

```
 acl static_path path_beg -i /static /images /javascript
 use_backend static_path_host if static_path
 default_backend default_web

backend static_path_host
 mode http
 server web1 10.0.0.201 check inter 2000 fall 3 rise 5

backend default_web
 mode http
 server web1 10.0.0.202:80 check inter 2000 fall 3 rise 5
```

3.8 根据请求参数匹配

```
frontend  main *:5000
 mode http
 balance roundrobin
 log global
 option httplog
 acl static_path urlp(username) -m str test
 use_backend static_path_host if static_path
 default_backend default_web

backend static_path_host
 mode http
 server web1 10.0.0.201 check inter 2000 fall 3 rise 5

backend default_web
 mode http
 server web1 10.0.0.202:80 check inter 2000 fall 3 rise 5
```

3.9 根据请求方式匹配

```
#示例 1：
frontend  main *:5000
 mode http
 balance roundrobin
 log global
 option httplog
 acl static_path path_end -i .php
 use_backend static_path_host if METH_POST static_path
 default_backend default_web

backend static_path_host
 mode http
 server web1 10.0.0.201 check inter 2000 fall 3 rise 5
```

```
backend default_web
 mode http
 server web1 10.0.0.202:80 check inter 2000 fall 3 rise 5

#示例 2：
frontend  main *:5000
 use_backend static_path_host if METH_POST
 default_backend default_web

backend static_path_host
 mode http
 server web1 10.0.0.201 check inter 2000 fall 3 rise 5

backend default_web
 mode http
 server web1 10.0.0.202:80 check inter 2000 fall 3 rise 5
```

3.10 根据是否 https 请求匹配

```
frontend  main *:5000
 acl https ssl_fc
 use_backend static_path_host if https
 default_backend default_web

backend static_path_host
 mode http
 server web1 10.0.0.201 check inter 2000 fall 3 rise 5

backend default_web
 mode http
 server web1 10.0.0.202:80 check inter 2000 fall 3 rise 5
```

3.11 根据 cookie 请求匹配

```
#cookie name=123 用户访问，其他用户将禁止
  acl url_static  cookie(name)  123
  use_backend static  if url_static
```

3.12 根据二进制数据匹配

```
# Match the string Hello at the beginning of the input stream
# (Hexadecimal values: x48 x65 x6c x6c x6f x0a)
#为了匹配无法安全地以字符串形式表示的二进制数据块，使用匹配类型 -m bin。
#这样做时，模式必须作为一系列十六进制数字传递，这些数字必须是偶数个。每对数字代表一个字节。十六进制数字可以是大写或小写。
acl hello payload(0,6) -m bin 48656c6c6f0a
```

3.13 拒绝访问示例

```
#拒绝指定文件中的 ip 列表以外的 IP 访问
frontend www
 bind :80
 acl safe_ip src -n -f /safelist.txt
 tcp-request content reject unless safe_ip

#拒绝 user-agent 包含 evil 字符串的访问，这有助于阻止恶意软件或爬虫的访问，它们可能使用特定的 User-Agent 标识自己。
frontend example
 bind :80
 http-request deny if { req.hdr(user-agent) -m sub evil }

#拒绝 user-agent 长度等于 32 的访问，由于攻击者有时会使用 32 字符长的随机 MD5 散列作为 User-Agent 来尝试逃避检测，此规则可以识别并阻止这类尝试
frontend example
 bind :80
 http-request deny if { req.hdr(user-agent) -m len 32 }

#拒绝 user-agent 长度小于 32 的访问，可以以此来过滤可能的非标准或恶意请求
frontend example
 bind :80
 http-request deny if { req.hdr(user-agent) -m len le 32 }

#拒绝对指定路径的访问
frontend example
 bind :80
 http-request deny if { path_beg /wp-admin/ }

#拒绝对隐藏文件的访问
frontend example
 bind :80
 http-request deny if { path -m sub /. }
```

3.14 status 响应码处理

```
acl status_404 status 404
http-response set-header Location https://sait.ru/ErrorPages/404_ru.html if status_404
http-response set-status 302 if status_404

acl status_404 status 500:511
http-response set-status 302 if status_404

acl status_5xx status 500 502 503 504
```

```
http-response set-header Location https://sait.ru/ErrorPages/503.html if status_5xx
http-response set-status 302 if status_5xx
```

3.15 根据响应 body_len 处理

```
frontend www
 bind :80
 acl 10kb_response res.body_len gt 10000    #响应 body 长度大于 10000bytes 时
 http-response set-status 302 if 10kb_response
```

3.16 payload 流量区分

payload 必须在 tcp 模式下才生效

3.16.1 http 示例

```
frontend  main
 bind *:5000
 mode tcp
 tcp-request inspect-delay 1ms




 acl is_http req.payload(0,3) -m bin 474554 504f53 505554 44454c 4f5054 484541 434f4e
545241

 tcp-request content accept if is_http

 use_backend static if is_http
```

3.16.2 完整示例

```
frontend  main
 bind *:5000
 mode tcp
 tcp-request inspect-delay 1ms
 acl is_http req.payload(0,3) -m bin 474554 504f53 505554 44454c 4f5054 484541 434f4e
545241
 acl is_ssh req.payload(0,3) -m bin 535348
 acl is_rdp req.payload(0,3) -m bin 030000
 acl is_https req.payload(0,3) -m bin 160301
 acl is_http req.payload(0,0) -m reg .*key.*
 acl is_http req.payload(0,0) -m reg "{.*\"key\":\"value\".*}"

# 设置四层允许通过
 tcp-request content accept if is_http
 tcp-request content accept if is_https
 tcp-request content accept if is_ssh
```

```
 tcp-request content accept if is_rdp
 tcp-request content accept

# 分发到对应的 backend
 use_backend http if is_http
 use_backend ss-out if !is_https
 use_backend ssh if is_ssh
 use_backend rdp if is_rdp
 use_backend socks5
```

3.17 HTTP 访问控制

```
listen web_host
 bind 192.168.32.204:80
 mode http
 balance roundrobin
 log global
 option httplog
 acl static_path path_beg -i /static /images /javascript
 use_backend static_path_host if static_path
 acl badguy_deny src 192.168.32.200
 http-request deny if badguy_deny
 http-request allow
 default_backend default_web

backend static_path_host
 mode http
 server web1 10.0.0.201 check inter 2000 fall 3 rise 5

backend default_web
 mode http
 server web1 10.0.0.202:80 check inter 2000 fall 3 rise 5
```

3.18 预定义 ACL 使用

预定义 ACL

```
ACL name              Equivalent to                  Usage
FALSE                 always_false                   never match
HTTP                  req_proto_http                 match if protocol is valid HTTP
HTTP_1.0              req_ver 1.0                    match HTTP version 1.0
HTTP_1.1              req_ver 1.1                    match HTTP version 1.1
HTTP_CONTENT          hdr_val(content-length) gt 0   match an existing content-length
HTTP_URL_ABS          url_reg ^[^/:]*://             match absolute URL with scheme
HTTP_URL_SLASH        url_beg /                      match URL beginning with "/"
HTTP_URL_STAR         url *                          match URL equal to "*"
```

```
LOCALHOST               src 127.0.0.1/8                 match connection from local host
METH_CONNECT            method CONNECT                  match HTTP CONNECT method
METH_DELETE             method DELETE                   match HTTP DELETE method
METH_GET                method GET HEAD                 match HTTP GET or HEAD method
METH_HEAD               method HEAD                     match HTTP HEAD method
METH_OPTIONS            method OPTIONS                  match HTTP OPTIONS method
METH_POST               method POST                     match HTTP POST method
METH_PUT                method PUT                      match HTTP PUT method
METH_TRACE              method TRACE                    match HTTP TRACE method
RDP_COOKIE              req_rdp_cookie_cnt gt 0         match presence of an RDP cookie
REQ_CONTENT             req_len gt 0                    match data in the request buffer
TRUE                    always_true                     always match
WAIT_END                wait_end                        wait for end of content analysis
```

预定义 ACL 使用

```
listen web_host
 bind 192.168.32.204:80
 mode http
 balance roundrobin
 log global
 option httplog
 acl static_path path_beg -i /static /images /javascript
 use_backend static_path_host if HTTP_1.1 TRUE static_path
 default_backend default_web

backend php_server_host
 mode http
 server web1 192.168.32.203 check inter 2000 fall 3 rise 5

backend static_path_host
 mode http
 server web1 10.0.0.201 check inter 2000 fall 3 rise 5

backend default_web
 mode http
 server web1 10.0.0.202:80 check inter 2000 fall 3 rise 5
```

3.19 常见的控制类型

```
#常见的控制类型
设置或修改请求头
作用：设置或修改请求中的头信息。
示例：
set-header X-Forwarded-For %[src]
```

添加请求头
作用：在请求中添加新的头信息。
示例：
add-header X-Client-IP %[src]

删除请求头
作用：从请求中删除指定的头信息。
示例：
del-header X-Cache

重定向请求
作用：将请求重定向到另一个 URL。
示例：
redirect location http://example.com/newpath if { some_condition }

跟踪会话状态
作用：跟踪和管理会话状态。
示例：
track-sc0 src table my_table

设置变量
作用：在请求处理过程中设置变量。
示例：
set-var(txn.is_admin) 1 if { path_beg /admin }

删除变量
作用：删除请求处理过程中的变量。
示例：
unset-var(txn.is_admin) if { path_beg /public }

设置防火墙标记
作用：设置或修改防火墙标记。
示例：
set-mark 0x100 if { src 192.168.1.0/24 }

设置日志级别
作用：调整请求的日志记录级别。
示例：
set-log-level warning if { path_beg /error }

捕获请求信息
作用：捕获请求中的某些信息并记录到日志中。
示例：
capture req.uri len 128

替换请求头中的值
作用：替换请求头中的某个值。
示例：
replace-value hdr(X-Real-IP) %[src] if { hdr(X-Real-IP) -m sub proxy }

替换整个请求头
作用：替换整个请求头。
示例：
replace-header Host example.com if { hdr(host) -m sub old.example.com }

允许请求通过
作用：允许请求通过，常用于访问控制。
示例：
allow if { src 192.168.1.0/24 }

将请求放入陷阱
作用：将请求放入陷阱，延迟响应。
示例：
tarpit if { src 192.168.1.0/24 }

修改请求路径
作用：改变请求的路径。
示例：
set-path /newpath%[path] if { path_beg /oldpath }

修改请求 URI
作用：改变请求的 URI。
示例：
set-uri /newuri%[uri] if { uri_beg /olduri }

修改请求查询字符串
作用：改变请求中的查询字符串。
示例：
set-query param1=value1&param2=value2 if { query -m sub oldparam }

重定向请求到指定前缀
作用：将请求重定向到指定的前缀。
示例：
redirect prefix http://example.com/newpath code 301 if { path_beg /oldpath }

重定向请求到指定 URL
作用：将请求重定向到指定的 URL。
示例：

redirect location http://example.com/newpage code 301 if { path_beg /oldpage }

重定向请求到指定协议
作用：将请求重定向到指定的协议（HTTP 或 HTTPS）。
示例：
redirect scheme https if { ssl_fc }

重定向请求并设置状态码
作用：将请求重定向并设置特定的 HTTP 状态码。
示例：
redirect status 404 if { path_beg /notfound }

拒绝请求
作用：拒绝请求，常用于访问控制。
示例：
reject if { src 192.168.1.0/24 }

发送客户端名称
作用：发送客户端的名称信息。
示例：
send-name if { src 192.168.1.0/24 }

发送客户端状态
作用：发送客户端的状态信息。
示例：
send-state if { src 192.168.1.0/24 }

发送 PROXY 协议头部
作用：发送 PROXY 协议头部，用于传递客户端的真实 IP 地址。
示例：
send-proxy if { src 192.168.1.0/24 }

发送 PROXY v2 协议头部
作用：发送 PROXY v2 协议头部，用于传递更详细的客户端信息。
示例：
send-proxy-v2 if { src 192.168.1.0/24 }

使用映射文件设置变量
作用：根据映射文件设置变量。
示例：
set-map(txn.user_role) hdr(User-Role) /etc/haproxy/user_roles.map

使用映射文件设置变量并处理结尾
作用：根据映射文件设置变量，并处理结尾。

示例：
set-map-end(txn.user_role) hdr(User-Role) /etc/haproxy/user_roles.map

使用映射文件设置变量并处理子串
作用：根据映射文件设置变量，并处理子串。
示例：
set-map-sub(txn.user_role) hdr(User-Role) /etc/haproxy/user_roles.map