方法一：COOKIE 方式

1：配置示例：

```
backend app
   balance     roundrobin




   cookie SERVERID insert indirect nocache
   server  app1 192.168.1.202:80 cookie server01 check inter 1000 fall 10
   server  app2 192.168.1.96:80 cookie server02 check inter 1000 fall 10
```

2：配置详解：

常用：

name：cookie 的 key 名称，用于实现持久连接
insert|prefix|rewrite： insert 最常用，其他没特殊需求不推荐使用
indirect：不会向后端服务器发送 HAproxy 设置的 cookie 信息，对于后端完全是透明的 #rewrite 模式下 cookie 必须要传递到后端服务器，不能设置 indirect 参数
nocache：当 client 和 hapoxy 之间有缓存时，不缓存 cookie

使用格式：

cookie <name> [rewrite|insert|prefix] [indirect] [nocache] [postonly] [preserve] [httponly] [secure] [domain <domain>]* [maxidle <idle>] [maxlife <life>]

参数详解：

prefix：

使用场景：在一些特殊环境下，客户端不支持多个"Set-Cookie"字段，这时可以使用 prefix。

参数详解：

haproxy 将在已存在的 cookie 上添加前缀 cookie 值，这个前缀部分是 server 指令中的 cookie 设置的，代表的是服务端标识符。

在客户端再次访问时，haproxy 将会自动移除这部分前缀，使得服务端只能看到它自己发出的 cookie。

使用 prefix 的时候，cookie 指令设置的 cookie 名必须和已存在的 cookie 一样(在本文的环境中是 PHPSESSID)，否则 prefix 模式下的 haproxy 不会对响应报文做任何改变。

使用示例：

```
cookie PHPSESSID prefix indirect nocache
server app1 192.168.100.60:80 cookie app_server1
server app2 192.168.100.61:80 cookie app_server2
```

rewrite：

使用场景：需要一个单独 cookie 专门控制会话保持，防止 haproxy 自定义的 cookie 覆盖已有的 cookie，主要在后端 cookie 特别多的情况下，不常见。

参数详解：

当后端服务器设置了 cookie 时，使用 rewrite 模式时，haproxy 将重写该 cookie 的值为后端服务器的标识符。当应用程序需要同时考虑"Set-Cookie"和"Cache-control"字段时，

该模式非常方便，因为应用程序可以决定是否应该设置一个为了保持会话的 cookie。

同样，rewrite 模式下的 haproxy 设置的 cookie 必须和后端服务器设置的 cookie 名称一致，否则不会做任何改变。

使用示例：

```
backend dynamic_group
cookie PHPSESSID rewrite nocache  #不能带 indirect 参数
server app1 192.168.100.60:80 cookie app_server1
server app2 192.168.100.61:80 cookie app_server2
```

postonly：仅在 POST 请求中插入 Cookie，这在某些特定场景下有用，但不常见于会话保持。

preserve：保留原始 Cookie 的属性，如 HTTPOnly 和 Secure 标志，当 HAProxy 需要修改或添加 Cookie 时使用。

httponly：设置 Cookie 为 HTTPOnly，增加安全性，防止 JavaScript 访问。

secure：指示 Cookie 仅通过 HTTPS 发送，增强安全性。

domain*：指定 Cookie 的有效域，允许跨子域名共享。

maxidle：定义 Cookie 在无活动状态下的最大存活时间（单位通常是秒），超过这个时间后，如果客户端再次发起请求，可能会重新分配服务器。

maxlife：设置 Cookie 的最大生命周期，控制 Cookie 多久后失效，帮助控制会话的持久性范围。

3：COOKIE 的作用域

defaults	frontend	listen	backend
yes	no	yes	yes

4：使用 cookie 实现会话保持以及如何忽略会话保持

其实，通过 cookie 表保持和后端的会话只是默认情况，haproxy 允许"即使使用了 cookie 也不进行会话绑定"的功能，这可以通过 ignore-persist 指令来实现。当满足该指令的要求时，表示不将该 cookie 插入到 cookie 表中，因此无法实现会话保持，即使 haproxy 设置了 cookie 也没用。

例如，在 backend 中指定如下配置：

```
backend dynamic_group
  acl url_static  path_beg       /static /images /img /css
  acl url_static  path_end       .gif .png .jpg .css .js
  ignore-persist  if url_static
```

```
  cookie app_cook insert nocache
  server app1 192.168.100.60:80 cookie app_server1
  server app2 192.168.100.61:80 cookie app_server2
```

这表示当请求 uri 以".jpg"结尾时，将忽略会话保持功能。这表示，对于 jpg 结尾的请求，app_cook 这个 cookie 从头到尾都是摆设。

方法二：stick table（推荐)

用法：

type ip|integer|string：使用什么类型的 key 作为客户端标识符。可以是客户端的源 IP，可以是一个整数 ID 值，也可以是一段从请求报文或响应报文中匹配出来的字符串。

size：表中允许的最大 stickiness 记录数量。单位使用 k、m 和 g 表示，分别表示 1024、2^20 和 2^30 条记录。

expire：stickiness 记录的过期时长。当某记录被操作后，过了一段时间就会过期，过期的记录会自动从 stick table 中移除，释放表空间。

nopurge：默认情况下，当表满后，如果还有新的 stickiness 记录要插入进来，haproxy 会自动将一部分老旧的 stickiness 记录 flush 掉，以释放空间存储新纪录。指定 nopurge 后，将不进行 flush，只能通过记录过期来释放表空间，因此该选项必须配合 expire 选项同时使用。

peers：指定要将 stick table 中的记录 replication 到对端 haproxy 节点。

store：指定要存储在 stick table 中的额外状态统计数据。其中代表后端服务器的标识符 server ID(即 key/value 的 value 部分)会自动插入，无需显式指定。

11

1k1 x 210 = 1024

1m1 x 220 = 1048576

1g1 x 230 = 1073741824

1：使用客户端源 IP 作为客户端标识符

```
frontend http-in
  bind          *:80
  mode           http
  log          global

  acl url_static   path_beg  -i /static /images /stylesheets
  acl url_static   path_end  -i .jpg .jpeg .gif .png .ico .bmp .html

  use_backend     static_group   if url_static
  default_backend  dynamic_group

backend dynamic_group
  stick-table type ip size 5k expire 1m  #秒 (s)、分钟(m)、小时(h)、天(d)
  stick on src
  balance roundrobin
```

```
  server app1 192.168.100.60:80 check rise 1 maxconn 3000
  server app2 192.168.100.61:80 check rise 1 maxconn 3000

backend static_group
  stick-table type ip size 5k expire 1m
  stick on src
  balance          roundrobin

  server staticsrv1  192.168.100.62:80 check rise 1 maxconn 5000
  server staticsrv2  192.168.100.63:80 check rise 1 maxconn 5000
```

2：使用 cookie 作为客户端标识符

```
backend dynamic_group
  stick-table type string len 32 size 5k expire 2m
  stick on req.cook(PHPSESSID)
  stick store-response res.cook(PHPSESSID)
  balance roundrobin

  server app1 192.168.100.60:80 check rise 1 maxconn 3000
  server app2 192.168.100.61:80 check rise 1 maxconn 3000
```

3：使用 string 作为客户端标识符

```
backend dynamic_group
  stick-table type string size 5k expire 2m
  stick on req.hdr(Host)
  balance roundrobin

  server app1 192.168.100.60:80 check rise 1 maxconn 3000
  server app2 192.168.100.61:80 check rise 1 maxconn 3000
```