



# **Relatório - Balanceamento de carga em Servidores Web com HAProxy e Keepalived**

Disponibilidade e Desempenho

2021 - 2022

**Bruno Teixeira**  
a2019100036@isec.pt

# Conteúdo

<b>1</b>	<b>Introdução</b>	<b>1</b>
<b>2</b>	<b>Desenvolvimento</b>	<b>2</b>
2.1	Descrição das tecnologias usadas . . . . .	2
2.2	Flask, MariaDB e Galera Cluster . . . . .	2
2.3	HAProxy . . . . .	2
2.4	KeepAlived . . . . .	2
2.5	Ambiente para as experiências . . . . .	3
2.5.1	Aplicação Web . . . . .	3
2.5.2	Configuração do HAProxy . . . . .	4
2.5.3	Configuração do KeepAlived . . . . .	5
2.6	Experiências . . . . .	6
2.6.1	Experiência 01 . . . . .	6
2.6.2	Experiência 02 . . . . .	8
2.6.3	Experiência 03 . . . . .	13
2.6.4	Experiência 04 . . . . .	16
<b>3</b>	<b>Conclusão</b>	<b>20</b>

# Lista de Figuras

2.1	Index - Aplicação Web . . . . .	3
2.2	Carrinho - Aplicação Web . . . . .	4
2.3	Configuração - HAProxy . . . . .	5
2.4	Estatísticas - HAProxy . . . . .	5
2.5	Configuração - Keepalived . . . . .	6
2.6	Esquema - Experiência 01 . . . . .	7
2.7	Wireshark - Experiência 01 . . . . .	7
2.8	Single Point of Failure - Experiência 01 . . . . .	8
2.9	Esquema - Experiência 02 . . . . .	8
2.10	Esquema com <i>fail-over</i> - Experiência 02 . . . . .	9
2.11	Wireshark no 192.168.1.183 - Experiência 02 . . . . .	9
2.12	Estado inicial do KeepAlived em ambos os servidores - Experiência 02 . . . . .	10
2.13	Wireshark no 192.168.1.183 com o HAProxy desligado - Experiência 02 . . . . .	10
2.14	Estado atual do KeepAlived em ambos os servidores - Experiência 02 . . . . .	11
2.15	Wireshark no 192.168.1.183 com o HAProxy retomado - Experiência 02 . . . . .	11
2.16	Estado final do KeepAlived - Experiência 02 . . . . .	12
2.17	Single Point of Failure - Experiência 02 . . . . .	12
2.18	Configuração do <i>Galera Cluster</i> no mariadb01 - Experiência 03 . . . . .	13
2.19	Configuração do <i>Galera Cluster</i> no mariadb02 - Experiência 03 . . . . .	13
2.20	Tamanho do <i>Galera Cluster</i> - Experiência 03 . . . . .	14
2.21	Configuração do HAProxy - Experiência 03 . . . . .	14
2.22	Esquema - Experiência 03 . . . . .	14
2.23	Primeira conexão - Experiência 03 . . . . .	15
2.24	Atualizada a página da aplicação - Experiência 03 . . . . .	15
2.25	<i>Query SQL</i> depois de atualizada a página da aplicação - Experiência 03 . . . . .	15
2.26	Single Point of Failure - Experiência 03 . . . . .	16
2.27	Configuração do HAProxy - Experiência 04 . . . . .	16
2.28	Configuração do KeepAlived - Experiência 04 . . . . .	17
2.29	Variável de ambiente DB_HOST - Experiência 04 . . . . .	17
2.30	Esquema - Experiência 04 . . . . .	17
2.31	Esquema com <i>failover</i> - Experiência 04 . . . . .	18
2.32	<i>Announcements</i> do HAProxy01 e HAProxy03 - Experiência 04 . . . . .	18
2.33	Troca de <i>MASTER</i> e <i>BACKUP</i> - Experiência 04 . . . . .	18
2.34	Reativado serviço HAProxy em HAProxy03 - Experiência 04 . . . . .	19

# Capítulo 1

## Introdução

Este trabalho tem como objetivo estudar o balanceamento de carga e o *failover* em servidores *web* com o HAProxy e o KeepAlived.

O principal foco é, criar alguns cenários com possíveis falhas, analisar esses cenários, descobrir os pontos fracos e tentar sempre minimizar o *downtime*.

Para alcançar este objetivo foram então criadas algumas experiências de modo a perceber como é possível criar uma infraestrutura segura e com um *downtime* reduzido ou até mesmo nulo aplicando ao mesmo tempo o conceito de balanceamento de carga.

## Capítulo 2

# Desenvolvimento

### 2.1 Descrição das tecnologias usadas

### 2.2 Flask, MariaDB e Galera Cluster

O Flask é um *micro-framework* do *Python* destinado principalmente para pequenas aplicações com requisitos mais simples. O mesmo funciona bastante bem com bases de dados tendo sido este o escolhido para o desenvolvimento da aplicação *web* para este trabalho.

Para que a aplicação fosse *stateful* foi então usada uma base de dados em MariaDB, uma vez que a mesma tem uma comunidade enorme na Internet, tornando-a bastante simples de ser utilizada.

O *Galera Cluster* é um *cluster* virtual para MariaDB que permite a replicação entre diferentes bases de dados, mantendo assim uma disponibilidade e desempenho alto uma vez que existe mais do que uma base de dados para responder a diferentes *queries* dos clientes, sendo que todas as mudanças feitas numa base de dados são replicadas para a outra.

### 2.3 HAProxy

O HAProxy é um serviço Linux que garante o balanceamento de carga para HTTP e TCP. Na prática, o mesmo recebe as conexões dos utilizadores e atua como *proxy*, criando um canal de comunicação entre o utilizador e um dos *webserver*s.

O HAProxy funciona em dois modos diferentes, HTTP ou TCP.

- Quando opera em TCP dizemos que é um *Proxy* de camada 4 (OSI)
  - Quando o HAProxy opera neste modo, o mesmo apenas tem acesso a qual IP e Porto o cliente está a tentar aceder, não conseguindo assim ver a informação trocada entre de ambas as partes.
- Quando opera em HTTP dizemos que é um *Proxy* de camada 7 (OSI)
  - Quando o HAProxy opera neste modo, o mesmo tem acesso a toda a informação, logo estamos a confiar no mesmo para ter acesso a esses dados, dados que transitam de um lado para o outro.

### 2.4 KeepAlived

O objetivo principal do KeepAlived é fornecer instalações simples para existir balanceamento de carga e alta disponibilidade (a alta disponibilidade é alcançada pelo protocolo VRRP) para sistemas baseados em Linux. O Keepalived agrega então um conjunto de servidores de balanceamento de carga (HAProxy), e consoante a saúde dos mesmos, ele toma decisões sobre pelo qual deverá passar a operacionalização.

## 2.5 Ambiente para as experiências

Para serem feitas algumas experiências foi criado um ambiente com várias máquinas virtuais, estando estas agregadas a um virtualizador ESXi, ou seja, todas as máquinas estão na mesma LAN.

- **webserver01** - 192.168.1.180
- **webserver02** - 192.168.1.181
- **mariadb01** - 192.168.1.182
- **mariadb02** - 192.168.1.186
- **haproxy01** - 192.168.1.183
- **haproxy02** - 192.168.1.184
- **haproxy03** - 192.168.1.185
- **haproxy04** - 192.168.1.187

### 2.5.1 Aplicação Web

Como descrito anteriormente, foi criada uma aplicação em Flask. Esta aplicação funciona como uma espécie de lista de compras em que o utilizador depois de fazer o *login*, consegue adicionar e eliminar produtos do seu cesto.



Figura 2.1: Index - Aplicação Web

Lista de Compras
Logout Carrinho

ID	Tipo	Quantidade	Local de Compra	
1	Cenoura	3	Pingo Doce	Apagar
2	Alface	1	Pingo Doce	Apagar
3	Laranjas	5	Pingo Doce	Apagar

Carrinho

Tipo

Quantidade

Local de Compra

Inserir

Figura 2.2: Carrinho - Aplicação Web

### 2.5.2 Configuração do HAProxy

No ficheiro de configuração do HAProxy (localizado em `/etc/haproxy/haproxy.cfg`) existem 5 secções, sendo que estas definem como é que o servidor se comporta, quais são as definições por omissão, e como é que o cliente faz pedidos e recebe respostas.

- **global**
  - Nesta primeira secção estão definidas as medidas em que o processo vai operar, sendo estas medidas de um nível mais baixo, ou seja, relacionadas com o sistema operativo.
- **defaults**
  - Esta secção não é obrigatória, no entanto permite reduzir a duplicação de comandos, uma vez que as configurações feitas aqui são aplicadas na secção **frontend** e **backend**.
- **listen**
  - Aqui podemos combinar o **frontend** e **backend** ao mesmo tempo. Isto é útil, pois é aqui feito o redirecionamento para o **endpoint** de estatísticas.
- **frontend**
  - Nesta secção definimos como é que os pedidos dos utilizadores irão ser encaminhados para o **backend**.
- **backend**
  - Aqui definimos os **webserver**s que vão operar na infraestrutura, definindo também o algoritmo de **load balancing** a ser utilizado

```

1 # O comando "stats socket" ativa a API do HAProxy sendo assim possível gerar um "endpoint" com todas as estatísticas do proxy e dos servidores web
2 global
3     stats socket /run/haproxy/admin.sock mode 660 level admin
4
5 # Neste caso usamos o modo "http"
6 defaults
7     mode http
8
9 # Configuração referente ao "endpoint" de estatísticas
10 # Por motivos de simplicidade o "auth" está sem encriptação
11 listen stats
12     bind 192.168.1.183:9999
13     stats enable
14     stats hide-version
15     stats refresh 10s
16     stats url /stats
17     stats auth haproxy:haproxy
18
19 # "http-in" é apenas um nome para o frontend
20 # Qualquer pessoa que se conecte a este servidor irá ser redirecionado para os "webserver" e fazemos isso usando "default_backend <nome do backend>"
21 frontend http-in
22     bind 192.168.1.183:80
23     default_backend webserver
24
25 # Damos o mesmo nome ao backend que demos no "default_backend"
26 # No "balance" escolhemos o algoritmo que quisermos utilizar na atribuição de "webserver" aos clientes (round-robin ou leastconn)
27 # Fazemos um "check" com um intervalo de 500 milissegundos, se falhar 3 vezes retira esse "webserver" do grupo
28 backend webserver
29     balance roundrobin
30     server webserver01 192.168.1.180:5000 check inter 500 fall 3
31     server webserver02 192.168.1.181:5000 check inter 500 fall 3

```

Figura 2.3: Configuração - HAProxy

Conforme foi configurado o HAProxy, ao acedermos a ***http://192.168.1.183:9999/stats*** conseguimos visualizar uma página *web* com várias estatísticas sobre os *webserver*.

## HAProxy

### Statistics Report for pid 1043

> General process information

pid = 1043 (process #1, nproc = 1, nbthread = 4)  
uptime = 0d 17h43m30s  
system limits: memmax = unlimited; ulimit-n = 1023  
maxsock = 1023; maxconn = 487; maxpipes = 0  
current conns = 1; current pipes = 0/0; conn rate = 0/sec; bit rate = 0.000 kbps  
Running tasks: 1/23; idle = 100 %

active UP  
active UP, going down  
active DOWN, going up  
active or backup DOWN  
backup UP  
backup UP, going down  
backup DOWN, going up  
not checked  
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  
• Disable refresh  
• Refresh now  
• CSV export

External resources:  
• [Primary site](#)  
• [Updates \(v2.0\)](#)  
• [Online manual](#)

stats

	Queue			Session rate			Sessions				Bytes		Denied		Errors		Warnings		Server												
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle	
Frontend				0	2	-	1	2	487	6			10 908	530 386	0	0	0	0	0	0	0	OPEN			0	0	0	0	0		
Backend	0	0		0	2		0	1	49	3	0	0s	10 908	530 386	0	0	0	3	0	0	0	17h43m UP		0	0	0	0	0			

http-in

	Queue			Session rate			Sessions				Bytes		Denied		Errors		Warnings		Server												
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle	
Frontend				0	1	-	0	3	487	6			17 152	180 658	0	0	0	0	0	0	0	OPEN									

webserver

	Queue			Session rate			Sessions				Bytes		Denied		Errors		Warnings		Server											
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
webserver01	0	0	-	0	3		0	2	-	29	29	3h14m	14 070	157 166	0	0	0	0	0	0	0	4m56s UP	L4OK in 0ms	1	Y	-	7	3	0s	-
webserver02	0	0	-	0	2		0	1	-	10	10	17h12m	3 082	23 492	0	0	0	0	0	0	0	4m38s UP	L4OK in 0ms	1	Y	-	4	2	0s	-
Backend	0	0		0	3		0	2	49	39	39	3h14m	17 152	180 658	0	0	0	0	0	0	0	4m56s UP		2	2	0	3	15h19m		

Figura 2.4: Estatísticas - HAProxy

## 2.5.3 Configuração do KeepAlived

No ficheiro de configuração do Keepalived (localizado em */etc/keepalived/keepalived.conf*), foi criado um **vrrp\_script** com o intuito de verificar, com um intervalo de 2 em 2 milissegundos, se o haproxy está a funcionar corretamente. Se o mesmo não estiver a funcionar, o peso dele diminui em 10 reduzindo assim a sua prioridade tornando o BACKUP num MASTER.

Depois disto, foi criado um **vrrp\_instance** que define uma instância individual do protocolo VRRP com alguns atributos.



```

1 # Define definições globais do processo
2 global_defs{
3     enable_script_security
4     script_user root
5 }
6
7 # Script que verifica o estado do HAProxy
8 vrrp_script check_haproxy {
9     script "service haproxy status"
10    interval 1
11    weight -10
12 }
13
14 # Instancia que executa o script, nomeando o HAProxy01 (192.168.1.183) como MASTER
15 # Escolhemos a interface que queremos usar
16 # O estado inicial
17 # Identificação do virtual_router_id sendo que esta tem de ser a mesma no BACKUP
18 # Prioridade do servidor, esta será anunciada no grupo VRRP
19 # IP virtual do grupo VRRP
20 vrrp_instance V1_1{
21     interface ens160
22     state MASTER
23     virtual_router_id 11
24     priority 101
25     authentication {
26         auth_type PASS
27         auth_pass algumacoisamuitocomplicada
28     }
29     virtual_ipaddress {
30         192.168.1.200
31     }
32     track_script {
33         check_haproxy
34     }
35 }

```

Figura 2.5: Configuração - Keepalived

O mesmo foi feito para o segundo servidor de HAProxy, no entanto foi alterada a prioridade e o estado para definir que este seria o BACKUP.

Foi também necessário alterar o Ip, para onde fazíamos *bind* inicialmente (**Ip do servidor HAProxy**), para o novo IP virtual (**192.168.1.200**) na secção de *frontend* do ficheiro de configuração do HAProxy.

Por fim foi preciso colocar *net.ipv4.ip\_nonlocal\_bind=1* no ficheiro */etc/sysctl.conf* uma vez que no segundo servidor de HAProxy o IP virtual ainda não está ativo (só fica ativo quando esse for o MASTER), logo não é possível iniciar o *bind*.

Esta configuração do KeepAlived apenas foi feita a partir da segunda experiência, uma vez que na primeira ainda não é usado o mesmo para mostrar os riscos que isso tem.

## 2.6 Experiências

### 2.6.1 Experiência 01

Nesta experiência foi usado um servidor de balanceamento de carga (HAProxy), dois *webservers* e uma base de dados externa (MariaDB). Foi feita uma captura no HAProxy para perceber o que acontecia quando o utilizador fazia um pedido ao mesmo.

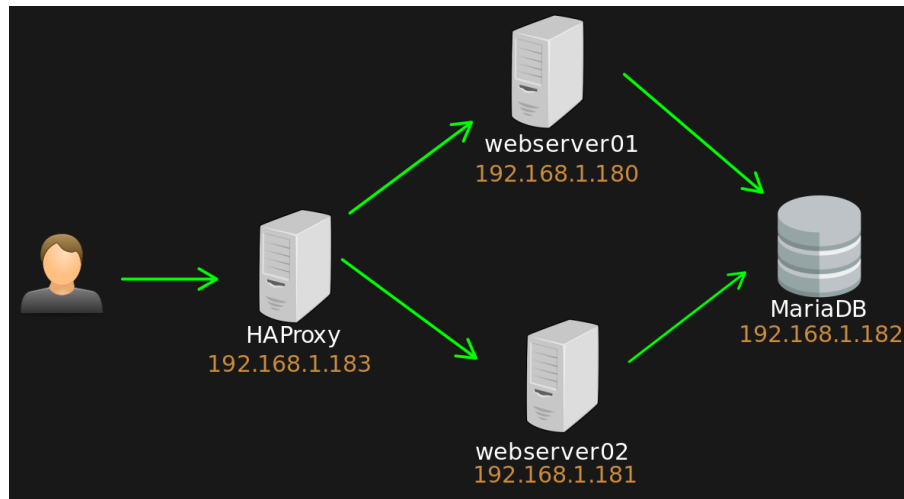


Figura 2.6: Esquema - Experiência 01

## Resultado

Conseguiu-se perceber que o cliente (192.168.1.123) envia um *HTTP GET Request* diretamente ao servidor HAProxy (192.168.1.183). De seguida, o servidor HAProxy faz um *HTTP GET Request* ao *webserver* disponível, que neste caso foi o *webserver01* (192.168.1.180). O *webserver01* responde com o *HTTP status code 200*, mostrando que a comunicação ocorreu sem falhas, sendo feito depois um redirecionamento do HAProxy para o cliente. Imediatamente a seguir foi feito outro pedido pelo mesmo cliente, no entanto consegue-se perceber que, como está a ser usar o algoritmo **round-robin**, quem respondeu foi o *webserver02*.

675	50.136854	192.168.1.123	192.168.1.183	HTTP	430 GET / HTTP/1.1	
679	50.137686	192.168.1.183	192.168.1.180	HTTP	406 GET / HTTP/1.1	
685	50.142710	192.168.1.180	192.168.1.183	HTTP	1327 HTTP/1.0 200 OK (text/html)	→ webservice01
690	50.142960	192.168.1.183	192.168.1.123	HTTP	1327 HTTP/1.0 200 OK (text/html)	
747	54.311420	192.168.1.123	192.168.1.183	HTTP	430 GET / HTTP/1.1	
751	54.312339	192.168.1.183	192.168.1.181	HTTP	406 GET / HTTP/1.1	
759	54.314710	192.168.1.181	192.168.1.183	HTTP	1184 HTTP/1.0 200 OK (text/html)	→ webservice02
762	54.314889	192.168.1.183	192.168.1.123	HTTP	2813 HTTP/1.0 200 OK (text/html)	

Figura 2.7: Wireshark - Experiência 01

## Problemas encontrados na Experiência 01

É perceptível que a experiência feita anteriormente tem alguns problemas, como a existência de dois SPOFs (Single Point of Failure).

- Existe um SPOF no HAProxy.
- Existe um SPOF na Base de Dados.

Ou seja, caso o servidor de HAProxy ou a base de dados deixe de operar, o cliente deixa de ter comunicação com os *webserver*s. Sabendo isto continuou-se com mais algumas experiências de modo a resolver estes problemas.

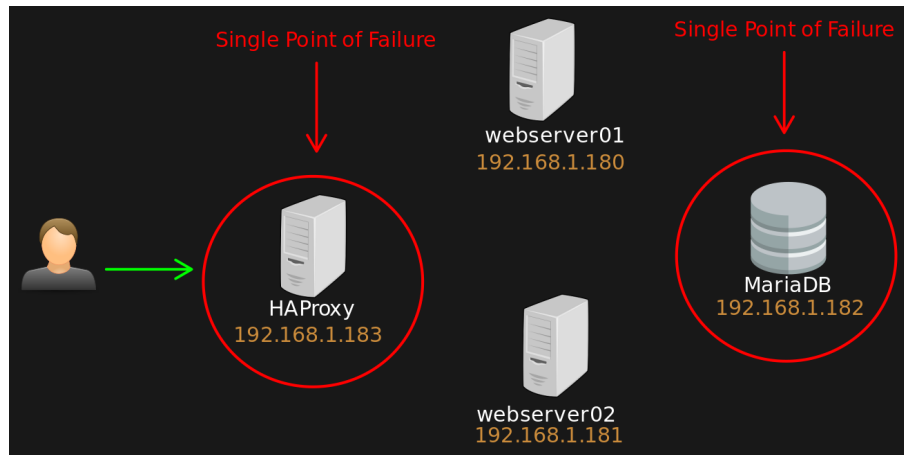


Figura 2.8: Single Point of Failure - Experiência 01

## 2.6.2 Experiência 02

Nesta segunda experiência apenas foi acrescentado mais um servidor de balanceamento de carga e o serviço de **KeepAlived** em ambos os servidores de HAProxy.

O objetivo da mesma era entender como seria feito o *fail-over* do **KeepAlived** e o que sucedia depois de um servidor *MASTER* tornar-se *BACKUP* e vice-versa.

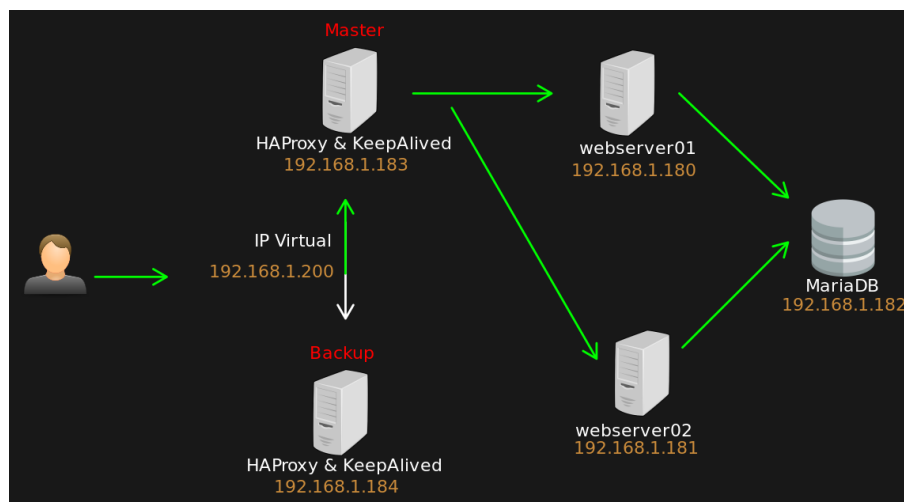


Figura 2.9: Esquema - Experiência 02

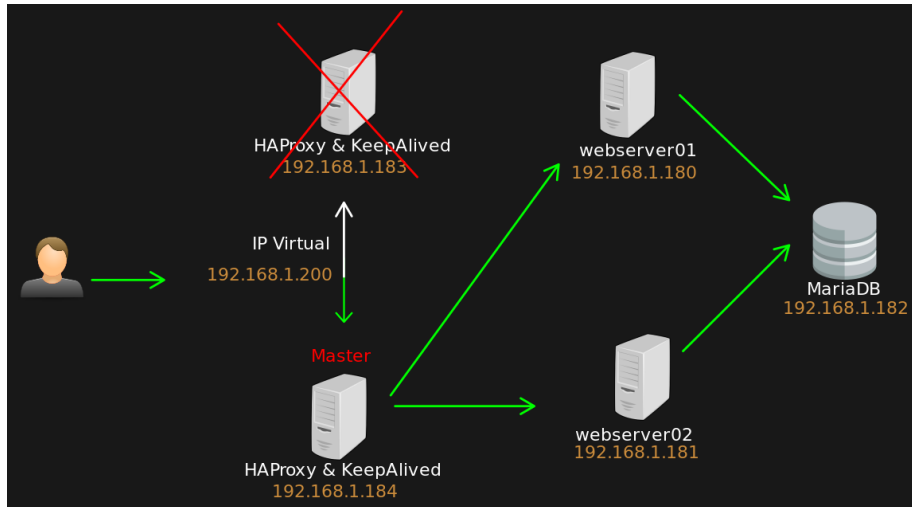


Figura 2.10: Esquema com *fail-over* - Experiência 02

## Resultado

Foram feitas várias capturas, tanto no **HAProxy01(MASTER)** como no **HAProxy02(BACKUP)** usando o *wireshark*.

Com a captura no **HAProxy01(192.168.1.183)** percebe-se que o mesmo emite, de segundo em segundo, um *announcement* dizendo a sua prioridade, que neste caso é 101. Isto acontece porque no protocolo VRRP apenas o *MASTER* emite mensagens estando os outros *BACKUPs* à escuta desse aviso.

238	17.795893	192.168.1.183	224.0.0.18	VRRP	54 Announc
251	18.796187	192.168.1.183	224.0.0.18	VRRP	54 Announc
264	19.796514	192.168.1.183	224.0.0.18	VRRP	54 Announc
279	20.796752	192.168.1.183	224.0.0.18	VRRP	54 Announc
292	21.796972	192.168.1.183	224.0.0.18	VRRP	54 Announc
305	22.797168	192.168.1.183	224.0.0.18	VRRP	54 Announc

Frame 238: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface /t	
Ethernet II, Src: VMware_db:73:4d (00:0c:29:db:73:4d), Dst: IPv4mcast_12 (01:00:5e:0	
Internet Protocol Version 4, Src: 192.168.1.183, Dst: 224.0.0.18	
Virtual Router Redundancy Protocol	
Version 2, Packet type 1 (Advertisement)	
Virtual Rtr ID: 11	
Priority: 101 (Non-default backup priority)	
Addr Count: 1	
Auth Type: Simple Text Authentication [RFC 2338] / Reserved [RFC 3768] (1)	
Adver Int: 1	
Checksum: 0x1ccf [correct]	

0010	00 28 44 1b 00 00 ff 70	d4 18 c0 a8 01 b7 e0 00	.(D...p .....
0020	00 12 21 0b 65 01 01 01	1c cf c0 a8 01 c8 61 6c	..!.E...al
0030	67 75 6d 61 63 6f		gumaco

Sending VRRP rout...rrp.prio, 1 byte   Packets: 1113 · Displayed: 83 (7.5%)   Profile: Default

Figura 2.11: Wireshark no 192.168.1.183 - Experiência 02

```

brun0@haproxy01:~$ sudo service keepalived status
● keepalived.service - Keepalive Daemon (LVS and VRRP)
   Loaded: loaded (/lib/systemd/system/keepalived.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2021-11-24 12:34:10 UTC; 5h 0min ago
     Main PID: 178585 (keepalived)
        Tasks: 2 (limit: 1070)
       Memory: 3.7M
      CGroup: /system.slice/keepalived.service
              └─178585 /usr/sbin/keepalived --dont-fork
                 └─178594 /usr/sbin/keepalived --dont-fork

Nov 24 17:24:52 haproxy01 Keepalived_vrrp[178594]: (V1_1) Changing effective priority from 101 to 91
Nov 24 17:24:54 haproxy01 Keepalived_vrrp[178594]: (V1_1) Master received advert from 192.168.1.184 with higher priority 100, ours 91
Nov 24 17:24:54 haproxy01 Keepalived_vrrp[178594]: (V1_1) Entering BACKUP STATE
Nov 24 17:29:51 haproxy01 Keepalived_vrrp[178594]: Script 'check_haproxy' now returning 0
Nov 24 17:29:51 haproxy01 Keepalived_vrrp[178594]: VRRP_Script(check_haproxy) succeeded
Nov 24 17:29:51 haproxy01 Keepalived_vrrp[178594]: (V1_1) Changing effective priority from 91 to 101
Nov 24 17:29:52 haproxy01 Keepalived_vrrp[178594]: (V1_1) received lower priority (100) advert from 192.168.1.184 - discarding
Nov 24 17:29:53 haproxy01 Keepalived_vrrp[178594]: (V1_1) received lower priority (100) advert from 192.168.1.184 - discarding
Nov 24 17:29:54 haproxy01 Keepalived_vrrp[178594]: (V1_1) received lower priority (100) advert from 192.168.1.184 - discarding
Nov 24 17:29:54 haproxy01 Keepalived_vrrp[178594]: (V1_1) Entering MASTER STATE
brun0@haproxy01:~$

brun0@haproxy02:~$ sudo service keepalived status
● keepalived.service - Keepalive Daemon (LVS and VRRP)
   Loaded: loaded (/lib/systemd/system/keepalived.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2021-11-24 13:30:54 UTC; 4h 3min ago
     Main PID: 847 (keepalived)
        Tasks: 2 (limit: 1070)
       Memory: 20.8M
      CGroup: /system.slice/keepalived.service
              └─847 /usr/sbin/keepalived --dont-fork
                 └─898 /usr/sbin/keepalived --dont-fork

Nov 24 17:19:38 haproxy02 Keepalived_vrrp[898]: (V1_1) received lower priority (91) advert from 192.168.1.183 - discarding
Nov 24 17:19:38 haproxy02 Keepalived_vrrp[898]: (V1_1) Entering MASTER STATE
Nov 24 17:20:25 haproxy02 Keepalived_vrrp[898]: (V1_1) Master received advert from 192.168.1.183 with higher priority 101, ours 100
Nov 24 17:20:25 haproxy02 Keepalived_vrrp[898]: (V1_1) Entering BACKUP STATE
Nov 24 17:24:52 haproxy02 Keepalived_vrrp[898]: (V1_1) received lower priority (91) advert from 192.168.1.183 - discarding
Nov 24 17:24:53 haproxy02 Keepalived_vrrp[898]: (V1_1) received lower priority (91) advert from 192.168.1.183 - discarding
Nov 24 17:24:54 haproxy02 Keepalived_vrrp[898]: (V1_1) received lower priority (91) advert from 192.168.1.183 - discarding
Nov 24 17:24:54 haproxy02 Keepalived_vrrp[898]: (V1_1) Entering MASTER STATE
Nov 24 17:29:54 haproxy02 Keepalived_vrrp[898]: (V1_1) Master received advert from 192.168.1.183 with higher priority 101, ours 100
Nov 24 17:29:54 haproxy02 Keepalived_vrrp[898]: (V1_1) Entering BACKUP STATE
brun0@haproxy02:~$

```

Figura 2.12: Estado inicial do KeepAlived em ambos os servidores - Experiência 02

Depois de desligar o serviço HAProxy do **HAProxy01(192.168.1.183)**, o mesmo fica com uma prioridade de 91 passando assim para o estado de *BACKUP* ao mesmo tempo que o **HAProxy02(192.168.1.184)** passa para o estado de *MASTER* uma vez que a sua prioridade é superior (100).

4179	312.859733	192.168.1.183	224.0.0.18	VRRP	54	Announc
4181	313.469517	192.168.1.184	224.0.0.18	VRRP	60	Announc
4186	314.469748	192.168.1.184	224.0.0.18	VRRP	60	Announc
4188	315.470003	192.168.1.184	224.0.0.18	VRRP	60	Announc
4189	316.470302	192.168.1.184	224.0.0.18	VRRP	60	Announc
4190	317.470524	192.168.1.184	224.0.0.18	VRRP	60	Announc
4196	318.470675	192.168.1.184	224.0.0.18	VRRP	60	Announc

▶ Frame 4179: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface /t  
 ▶ Ethernet II, Src: VMware\_db:73:4d (00:0c:29:db:73:4d), Dst: IPv4mcast\_12 (01:00:5e:00  
 ▶ Internet Protocol Version 4, Src: 192.168.1.183, Dst: 224.0.0.18  
 ▶ Virtual Router Redundancy Protocol  
   ▶ Version 2, Packet type 1 (Advertisement)  
   Virtual Rtr ID: 11  
   Priority: 91 (Non-default backup priority)  
   Addr Count: 1  
   Auth Type: Simple Text Authentication [RFC 2338] / Reserved [RFC 3768] (1)  
   Adver Int: 1  
   Checksum: 0x26cf [correct]  
   [Checksum Status: Good]  
   IP Address: 192.168.1.200  
   Authentication String: algumaco

Figura 2.13: Wireshark no 192.168.1.183 com o HAProxy desligado - Experiência 02

```

brun0@haproxy01:~$ sudo service keepalived status
● keepalived.service - Keepalive Daemon (LVS and VRRP)
   Loaded: loaded (/lib/systemd/system/keepalived.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2021-11-24 12:34:10 UTC; 5h 1min ago
     Main PID: 178585 (keepalived)
       Tasks: 2 (limit: 1070)
      Memory: 3.9M
     CGroup: /system.slice/keepalived.service
            └─178585 /usr/sbin/keepalived --dont-fork
              └─178594 /usr/sbin/keepalived --dont-fork

Nov 24 17:29:51 haproxy01 Keepalived_vrrp[178594]: (V1_1) Changing effective priority from 91 to 101
Nov 24 17:29:52 haproxy01 Keepalived_vrrp[178594]: (V1_1) received lower priority (100) advert from 192.168.1.184 - discarding
Nov 24 17:29:53 haproxy01 Keepalived_vrrp[178594]: (V1_1) received lower priority (100) advert from 192.168.1.184 - discarding
Nov 24 17:29:54 haproxy01 Keepalived_vrrp[178594]: (V1_1) received lower priority (100) advert from 192.168.1.184 - discarding
Nov 24 17:29:54 haproxy01 Keepalived_vrrp[178594]: (V1_1) Entering MASTER STATE
Nov 24 17:35:54 haproxy01 Keepalived_vrrp[178594]: Script 'check_haproxy' now returning 3
Nov 24 17:35:54 haproxy01 Keepalived_vrrp[178594]: VRRP_Script(check_haproxy) failed (exited with status 3)
Nov 24 17:35:54 haproxy01 Keepalived_vrrp[178594]: (V1_1) Changing effective priority from 101 to 91
Nov 24 17:35:57 haproxy01 Keepalived_vrrp[178594]: (V1_1) Master received advert from 192.168.1.184 with higher priority 100, ours 91
Nov 24 17:35:57 haproxy01 Keepalived_vrrp[178594]: (V1_1) Entering BACKUP STATE
brun0@haproxy01:~$

brun0@haproxy02:~$ sudo service keepalived status
● keepalived.service - Keepalive Daemon (LVS and VRRP)
   Loaded: loaded (/lib/systemd/system/keepalived.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2021-11-24 13:30:54 UTC; 4h 5min ago
     Main PID: 847 (keepalived)
       Tasks: 2 (limit: 1070)
      Memory: 20.9M
     CGroup: /system.slice/keepalived.service
            └─847 /usr/sbin/keepalived --dont-fork
              └─898 /usr/sbin/keepalived --dont-fork

Nov 24 17:24:52 haproxy02 Keepalived_vrrp[898]: (V1_1) received lower priority (91) advert from 192.168.1.183 - discarding
Nov 24 17:24:53 haproxy02 Keepalived_vrrp[898]: (V1_1) received lower priority (91) advert from 192.168.1.183 - discarding
Nov 24 17:24:54 haproxy02 Keepalived_vrrp[898]: (V1_1) received lower priority (91) advert from 192.168.1.183 - discarding
Nov 24 17:24:54 haproxy02 Keepalived_vrrp[898]: (V1_1) Entering MASTER STATE
Nov 24 17:29:54 haproxy02 Keepalived_vrrp[898]: (V1_1) Master received advert from 192.168.1.183 with higher priority 101, ours 100
Nov 24 17:29:54 haproxy02 Keepalived_vrrp[898]: (V1_1) Entering BACKUP STATE
Nov 24 17:35:54 haproxy02 Keepalived_vrrp[898]: (V1_1) received lower priority (91) advert from 192.168.1.183 - discarding
Nov 24 17:35:55 haproxy02 Keepalived_vrrp[898]: (V1_1) received lower priority (91) advert from 192.168.1.183 - discarding
Nov 24 17:35:56 haproxy02 Keepalived_vrrp[898]: (V1_1) received lower priority (91) advert from 192.168.1.183 - discarding
Nov 24 17:35:57 haproxy02 Keepalived_vrrp[898]: (V1_1) Entering MASTER STATE
brun0@haproxy02:~$

```

Figura 2.14: Estado atual do KeepAlived em ambos os servidores - Experiência 02

Para terminar, voltou-se a ativar o serviço haproxy no **HAProxy01(192.168.1.183)** tornando-se assim novamente *MASTER* uma vez que a preempção está ativa por omissão fazendo com que a sua prioridade volte a ser 101 como estava definida inicialmente.

4445	469.111033	192.168.1.183	224.0.0.18	VRRP	54	Announc
4458	470.111248	192.168.1.183	224.0.0.18	VRRP	54	Announc
4475	471.111472	192.168.1.183	224.0.0.18	VRRP	54	Announc
4488	472.111647	192.168.1.183	224.0.0.18	VRRP	54	Announc
4501	473.111860	192.168.1.183	224.0.0.18	VRRP	54	Announc
4520	474.112107	192.168.1.183	224.0.0.18	VRRP	54	Announc

▶ Frame 4445: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface /tr ▶ Ethernet II, Src: VMware_db:73:4d (00:0c:29:db:73:4d), Dst: IPv4mcast_12 (01:00:5e:00 ▶ Internet Protocol Version 4, Src: 192.168.1.183, Dst: 224.0.0.18 ▶ Virtual Router Redundancy Protocol ▶ Version 2, Packet type 1 (Advertisement) Virtual Rtr ID: 11 <b>Priority: 101 (Non-default backup priority)</b> Addr Count: 1 Auth Type: Simple Text Authentication [RFC 2338] / Reserved [RFC 3768] (1) Adver Int: 1 Checksum: 0x1ccf [correct] [Checksum Status: Good] IP Address: 192.168.1.200 Authentication String: algumaco
---

Figura 2.15: Wireshark no 192.168.1.183 com o HAProxy retomado - Experiência 02

```

brun0@haproxy01:~$ sudo service keepalived status
● keepalived.service - Keepalive Daemon (LVS and VRRP)
   Loaded: loaded (/lib/systemd/system/keepalived.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2021-11-24 12:34:10 UTC; 5h 4min ago
     Main PID: 178585 (keepalived)
       Tasks: 2 (limit: 1070)
        Memory: 3.7M
      CGroup: /system.slice/keepalived.service
              └─178585 /usr/sbin/keepalived --dont-fork
                 └─178594 /usr/sbin/keepalived --dont-fork

Nov 24 17:35:54 haproxy01 Keepalived_vrrp[178594]: (V1_1) Changing effective priority from 101 to 91
Nov 24 17:35:57 haproxy01 Keepalived_vrrp[178594]: (V1_1) Master received advert from 192.168.1.184 with higher priority 100, ours 91
Nov 24 17:35:57 haproxy01 Keepalived_vrrp[178594]: (V1_1) Entering BACKUP STATE
Nov 24 17:38:30 haproxy01 Keepalived_vrrp[178594]: Script 'check_haproxy' now returning 0
Nov 24 17:38:30 haproxy01 Keepalived_vrrp[178594]: VRRP_Script(check_haproxy) succeeded
Nov 24 17:38:30 haproxy01 Keepalived_vrrp[178594]: (V1_1) Changing effective priority from 91 to 101
Nov 24 17:38:30 haproxy01 Keepalived_vrrp[178594]: (V1_1) received lower priority (100) advert from 192.168.1.184 - discarding
Nov 24 17:38:31 haproxy01 Keepalived_vrrp[178594]: (V1_1) received lower priority (100) advert from 192.168.1.184 - discarding
Nov 24 17:38:32 haproxy01 Keepalived_vrrp[178594]: (V1_1) received lower priority (100) advert from 192.168.1.184 - discarding
Nov 24 17:38:32 haproxy01 Keepalived_vrrp[178594]: (V1_1) Entering MASTER STATE
brun0@haproxy01:~$

brun0@haproxy02: ~ 134x24
brun0@haproxy02:~$ sudo service keepalived status
● keepalived.service - Keepalive Daemon (LVS and VRRP)
   Loaded: loaded (/lib/systemd/system/keepalived.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2021-11-24 13:30:54 UTC; 4h 7min ago
     Main PID: 847 (keepalived)
       Tasks: 2 (limit: 1070)
        Memory: 20.8M
      CGroup: /system.slice/keepalived.service
              └─847 /usr/sbin/keepalived --dont-fork
                 └─898 /usr/sbin/keepalived --dont-fork

Nov 24 17:24:54 haproxy02 Keepalived_vrrp[898]: (V1_1) received lower priority (91) advert from 192.168.1.183 - discarding
Nov 24 17:24:54 haproxy02 Keepalived_vrrp[898]: (V1_1) Entering MASTER STATE
Nov 24 17:29:54 haproxy02 Keepalived_vrrp[898]: (V1_1) Master received advert from 192.168.1.183 with higher priority 101, ours 100
Nov 24 17:29:54 haproxy02 Keepalived_vrrp[898]: (V1_1) Entering BACKUP STATE
Nov 24 17:35:54 haproxy02 Keepalived_vrrp[898]: (V1_1) received lower priority (91) advert from 192.168.1.183 - discarding
Nov 24 17:35:55 haproxy02 Keepalived_vrrp[898]: (V1_1) received lower priority (91) advert from 192.168.1.183 - discarding
Nov 24 17:35:56 haproxy02 Keepalived_vrrp[898]: (V1_1) received lower priority (91) advert from 192.168.1.183 - discarding
Nov 24 17:35:57 haproxy02 Keepalived_vrrp[898]: (V1_1) Entering MASTER STATE
Nov 24 17:38:32 haproxy02 Keepalived_vrrp[898]: (V1_1) Master received advert from 192.168.1.183 with higher priority 101, ours 100
Nov 24 17:38:32 haproxy02 Keepalived_vrrp[898]: (V1_1) Entering BACKUP STATE
brun0@haproxy02:~$

```

Figura 2.16: Estado final do KeepAlived - Experiência 02

## Problemas encontrados na Experiência 02

Com esta nova arquitetura, foi possível resolver um SPOF(Single Point of Failure) colocando mais um servidor de balanceamento de carga e acrescentando o serviço de **KeepAlived**, no entanto é notório que continua a existir um SPOF na base de dados.

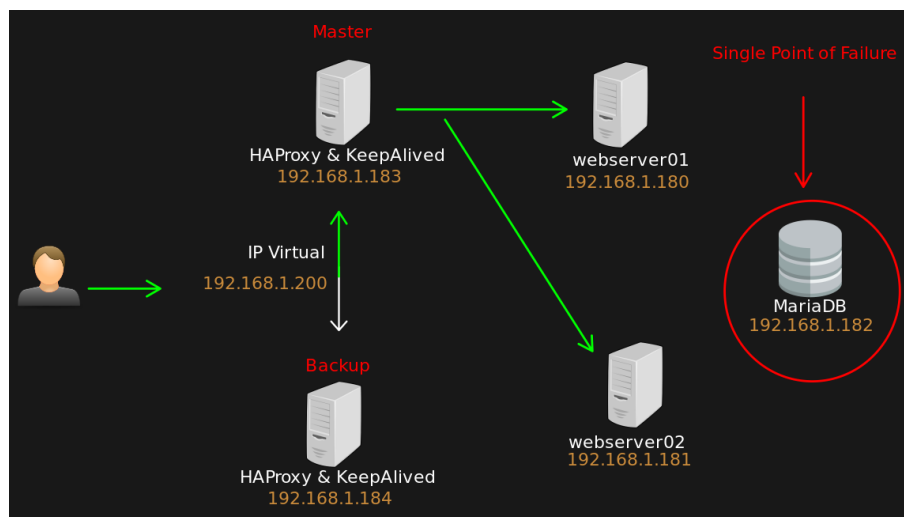


Figura 2.17: Single Point of Failure - Experiência 02

### 2.6.3 Experiência 03

Para resolver o SPOF encontrado na última experiência, foi preciso utilizar mais um servidor HA-Proxy(192.168.185) e mais uma base de dados(192.168.1.186). Esta base de dados agora vai-se juntar à base de dados (192.168.1.182) já existente fazendo assim um *Galera Cluster*.

No primeiro servidor de base de dados (192.168.1.182) foi criado o ficheiro **galera.cnf** localizado em **/etc/mysql/mariadb.conf.d/galera.cnf** contendo todas as configurações necessárias para a configuração do *cluster*.

```
1 [mysqld]
2 bind-address=0.0.0.0
3 default_storage_engine=InnoDB
4 binlog_format=row
5 innodb_autoinc_lock_mode=2
6 wsrep_on=ON
7 wsrep_provider=/usr/lib/galera/libgalera_smm.so
8
9 ;; IP da mariadb01 e da mariadb02
10 wsrep_cluster_address="gcomm://192.168.1.182,192.168.1.186"
11 wsrep_cluster_name="mariadb-galera-cluster"
12 wsrep_sst_method=rsync
13
14 ;; IP do servidor local
15 wsrep_node_address="192.168.1.182"
16 wsrep_node_name="galera-db-01"
```

Figura 2.18: Configuração do *Galera Cluster* no mariadb01 - Experiência 03

No segundo servidor de base de dados (192.168.1.186) foi também criado o ficheiro **galera.cnf** com as configurações necessárias ao mesmo fazer parte do *cluster*.

```
1 [mysqld]
2 bind-address=0.0.0.0
3 default_storage_engine=InnoDB
4 binlog_format=row
5 innodb_autoinc_lock_mode=2
6 wsrep_on=ON
7 wsrep_provider=/usr/lib/galera/libgalera_smm.so
8
9 ;; IP da mariadb01 e da mariadb02
10 wsrep_cluster_address="gcomm://192.168.1.182,192.168.1.186"
11 wsrep_cluster_name="mariadb-galera-cluster"
12 wsrep_sst_method=rsync
13
14 ;; IP do servidor local
15 wsrep_node_address="192.168.1.186"
16 wsrep_node_name="galera-db-01"
```

Figura 2.19: Configuração do *Galera Cluster* no mariadb02 - Experiência 03

Depois de ambos os servidores de base de dados configurados, foi visto que o tamanho do *cluster* era de 2, para isso foi utilizado o comando **sudo mysql -u root -p -e "show status like 'wsrep\_cluster\_size'"**.



```

brun0@mariadb01:~$ sudo mysql -u root -p -e "show status like 'wsrep_cluster_size'"
[sudo] password for brun0:
Enter password:
+-----+-----+
| Variable_name | Value |
+-----+-----+
| wsrep_cluster_size | 2      |
+-----+-----+

```

Figura 2.20: Tamanho do *Galera Cluster* - Experiência 03

De seguida o novo servidor de HAProxy (**192.168.1.185**) foi configurado de modo a receber e encaminhar tráfego dos clientes e das base de dados, fazendo assim um balanceamento de carga entre as bases de dados existentes no *Galera Cluster*.

Para isto, foi editado o ficheiro **haproxy.cfg** localizado em */etc/haproxy/haproxy.cfg*.

```

1 frontend mariadb_cluster_frontend
2     bind *:3306
3     mode tcp
4     option tcplog
5     default_backend galera_cluster_backend
6
7 ;; modo TCP
8 ;; utilizando roundrobin entre o mariadb01 e mariadb02
9 backend galera_cluster_backend
10    mode tcp
11    option tcpka
12    balance roundrobin
13    server mariadb01 192.168.1.182:3306 check weight 1
14    server mariadb02 192.168.1.186:3306 check weight 1

```

Figura 2.21: Configuração do HAProxy - Experiência 03

Por fim, foi necessário alterar o IP presente na variável de ambiente **DB\_HOST**, em ambos os *webservers*, de **192.168.1.182** para o IP do HAProxy (**192.168.1.185**).

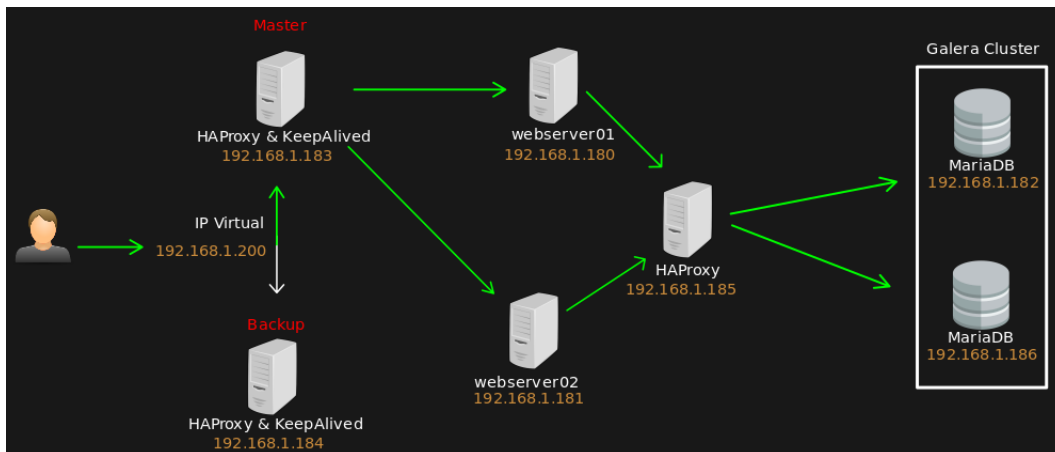


Figura 2.22: Esquema - Experiência 03

## Resultado

Foi realizada uma captura no HAProxy03 (**192.168.1.185**) usando o *wireshark* para perceber como é que o tráfego circulava na rede.

Inicialmente o *webserver01* emite um **ARP Request** para saber qual é o **MAC** do novo HAProxy (**192.168.1.185**), sendo que o HAProxy transmite o seu **MAC** e logo a seguir faz uma saudação (*Server Greeting*) à base de dados que neste caso foi à mariadb01 (**192.168.1.182**).

De seguida existe um *Login Request* e é perceptível que agora a comunicação entre *webserver* e base de dados, passa sempre pelo HAProxy.

103	24.100226	Vmware_e8:2f:8e	Broadcast	ARP	60 Who has 192.168.1.185? Tell 192.168.1.181
104	24.100293	Vmware_68:10:76	Vmware_e8:2f:8e	ARP	42 192.168.1.185 is at 00:0c:29:68:10:76
112	24.149549	192.168.1.182	192.168.1.185	MySQL	185 Server Greeting proto=10 version=10.4.22-MariaDB-1:10.4.22+maria-focal-log
114	24.149848	192.168.1.185	192.168.1.181	MySQL	185 Server Greeting proto=10 version=10.4.22-MariaDB-1:10.4.22+maria-focal-log
116	24.150565	192.168.1.181	192.168.1.185	MySQL	214 Login Request user=brun0 db=carrinho
118	24.150765	192.168.1.185	192.168.1.182	MySQL	214 Login Request user=brun0 db=carrinho
120	24.163095	192.168.1.182	192.168.1.185	MySQL	77 Response OK
122	24.163320	192.168.1.185	192.168.1.181	MySQL	77 Response OK
124	24.163698	192.168.1.181	192.168.1.185	MySQL	89 Request Query
126	24.163896	192.168.1.185	192.168.1.182	MySQL	89 Request Query
128	24.164471	192.168.1.182	192.168.1.185	MySQL	77 Response OK
130	24.164575	192.168.1.185	192.168.1.181	MySQL	77 Response OK

Figura 2.23: Primeira conexão - Experiência 03

Passado algum tempo, e já tendo sido feito o *login* na aplicação, foi atualizada a página do **carrinho** e mais uma vez aqui é notório o caminho da comunicação entre *webserver* e base de dados.

Desta vez o HAProxy encaminhou a *query* do cliente para a mariadb02 (**192.168.1.186**) e posteriormente esta respondeu ao HAProxy voltando a ser encaminhada essa resposta para o cliente.

8862	2300.471856	192.168.1.181	192.168.1.185	MySQL	272 Request Query
8863	2300.472034	192.168.1.185	192.168.1.186	MySQL	272 Request Query
8864	2300.472863	192.168.1.186	192.168.1.185	MySQL	416 Response
8866	2300.472986	192.168.1.185	192.168.1.181	MySQL	416 Response

Figura 2.24: Atualizada a página da aplicação - Experiência 03

- MySQL Protocol	
Packet Length: 234	
Packet Number: 0	
- Request Command Query	
Command: Query (3)	
Statement [truncated]: SELECT compras.id_compras AS id_compras, compras.tipo AS 'Tipo', compras.quantidade AS 'Quantidade', compras.local AS 'Local' \nFROM compras INNER JOIN clien-	
0000	00 0c 29 d0 f7 ca 00 0c 29 68 10 76 08 00 45 00 . . . . . ) h v . E
0010	01 22 02 56 40 00 40 06 52 bc c0 a8 01 b9 00 00 *bv@ 0 R . . . . .
0020	01 b5 ad d0 0c ea 64 a7 e5 62 37 e3 21 59 89 18 . . . . . d . b7 iv .
0030	01 f5 85 d0 00 00 01 01 08 0a f4 50 b5 af f9 63 . . . . . P . c
0040	2b 63 ea 00 00 00 03 53 45 4c 45 43 54 20 63 6f +c . . . . S ELECT co
0050	6d 76 72 61 73 2e 69 64 5f 63 6f 6d 70 72 61 73 mpras.id _compras
0060	20 41 53 20 69 64 5f 63 6f 6d 70 72 61 73 2c 20 AS id_c ompras,
0070	63 6f 6d 70 72 61 73 2e 74 69 70 6f 20 41 53 20 compras. tipo AS
0080	60 54 69 70 6f 60 2c 20 63 6f 6d 70 72 61 73 2e 'Tipo', compras.
0090	71 75 61 6e 74 69 64 61 64 65 60 20 41 53 20 60 51 quantidade de AS 'Q
00a0	75 61 6e 74 69 64 61 64 65 60 2c 20 63 6f 6d 70 uantidade e', comp
00b0	72 61 73 2e 6c 6f 63 61 6c 20 41 53 20 60 4c 6f ras. loca l AS 'Lo
00c0	63 61 6c 60 20 0a 46 52 4f 4d 20 63 6f 6d 70 72 cal' FR OM compr
00d0	61 73 20 49 4e 4e 45 52 20 4a 4f 49 4e 20 63 6c as INNER JOIN cli
00e0	69 65 6e 74 65 73 20 4f 4e 20 63 6f 6d 70 72 61 ientes O N compra
00f0	73 2e 69 64 5f 63 6c 69 65 6e 74 65 20 3d 20 63 s.id_cli ente = c
0100	6c 69 65 6e 74 65 73 20 69 64 5f 63 6c 69 65 6e lientes. id clien
0110	74 65 20 0a 57 4b 45 52 45 20 63 6f 6d 70 72 61 te \nWER E compra
0120	73 2e 69 64 5f 63 6c 69 65 6e 74 65 20 3d 20 37 s.id_cli ente = 7

Figura 2.25: Query SQL depois de atualizada a página da aplicação - Experiência 03

## Problemas encontrados na Experiência 03

Com a implementação do novo HAProxy e do *Galera Cluster*, foi possível resolver um SPOF que a antiga experiência tinha, no entanto passou a haver outro SPOF, agora no servidor de HAProxy que gere o *Galera Cluster*.

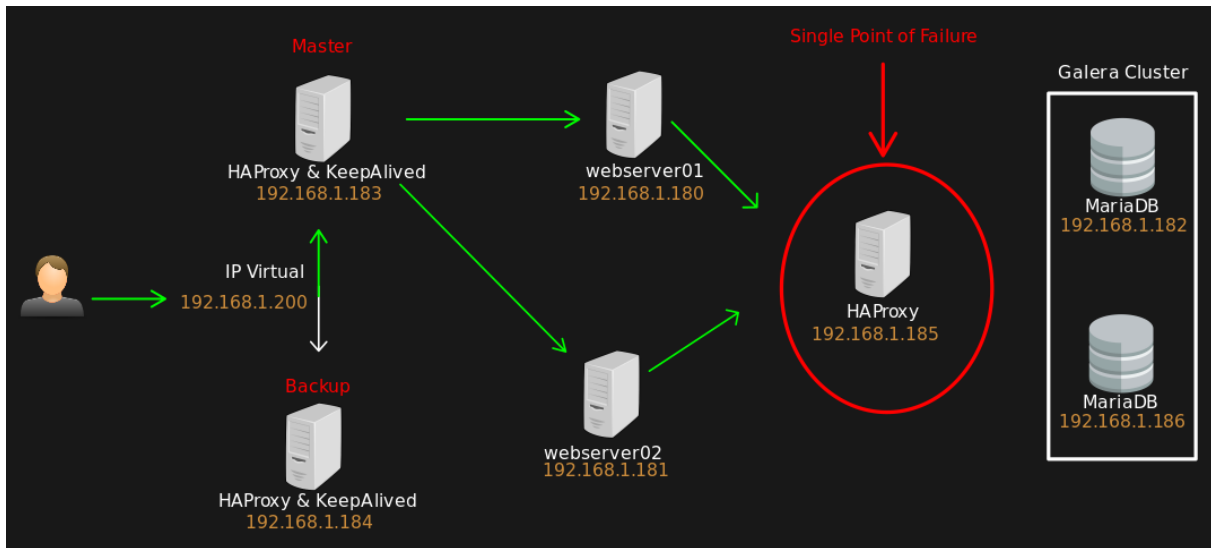


Figura 2.26: Single Point of Failure - Experiência 03

#### 2.6.4 Experiência 04

Para resolver o SPOF encontrado na última experiência, foi preciso utilizar mais um servidor de HAProxy (192.168.1.187) e acrescentar o KeepAlived em ambos os servidores HAProxy que fazem o balanceamento de carga para o *Galera Cluster*.

```

brun0@haproxy03:~$ sudo cat /etc/haproxy/haproxy.cfg
frontend mariadb_cluster_frontend
  bind *:3306
  mode tcp
  option tcplog
  default_backend galera_cluster_backend

backend galera_cluster_backend
  mode tcp
  option tcpka
  balance roundrobin
  server mysql-01 192.168.1.182:3306 check weight 1
  server mysql-02 192.168.1.186:3306 check weight 1

brun0@haproxy04:~$ sudo cat /etc/haproxy/haproxy.cfg
frontend mariadb_cluster_frontend
  bind *:3306
  mode tcp
  option tcplog
  default_backend galera_cluster_backend

backend galera_cluster_backend
  mode tcp
  option tcpka
  balance roundrobin
  server mysql-01 192.168.1.182:3306 check weight 1
  server mysql-02 192.168.1.186:3306 check weight 1

```

Figura 2.27: Configuração do HAProxy - Experiência 04

A configuração do KeepAlived foi exatamente igual, no entanto para estes dois servidores de HAProxy foi utilizado o ip virtual 192.168.1.250, tendo na mesma um *vrp\_script* que faz o controle da "vida" dos servidores de HAProxy.

```

brun0@haproxy03:~$ sudo cat /etc/keepalived/keepalived.conf
global_defs{
    enable_script_security
    script_user root
}

vrrp_script check_haproxy {
    script "service haproxy status"
    interval 1
    weight -10
}

vrrp_instance V1_1{
    interface ens160
    state MASTER
    virtual_router_id 11
    priority 101
    authentication {
        auth_type PASS
        auth_pass algumacoisamuitocomplicada
    }
    virtual_ipaddress {
        192.168.1.250
    }
    track_script {
        check_haproxy
    }
}

brun0@haproxy04:~$ sudo cat /etc/keepalived/keepalived.conf
global_defs{
    enable_script_security
    script_user root
}

vrrp_script check_haproxy {
    script "service haproxy status"
    interval 2
    weight -10
}

vrrp_instance V1_1{
    interface ens160
    state BACKUP
    virtual_router_id 11
    priority 100
    authentication {
        auth_type PASS
        auth_pass algumacoisamuitocomplicada
    }
    virtual_ipaddress {
        192.168.1.250
    }
    track_script {
        check_haproxy
    }
}

```

Figura 2.28: Configuração do KeepAlived - Experiência 04

Por fim, foi necessário alterar o IP presente na variável de ambiente `DB_HOST`, em ambos os *webserver*s, de `192.168.1.185` (IP do HAProxy03) para o IP virtual (`192.168.1.185`).

```

brun0@webserver01:~/repo/flask_app$ printenv | grep "DB_HOST"
DB_HOST=mysql+pymysql://brun0:toor@192.168.1.250:3306/carrinho
brun0@webserver02:~/repo/flask_app$ printenv | grep "DB_HOST"
DB_HOST=mysql+pymysql://brun0:toor@192.168.1.250:3306/carrinho

```

Figura 2.29: Variável de ambiente `DB_HOST` - Experiência 04

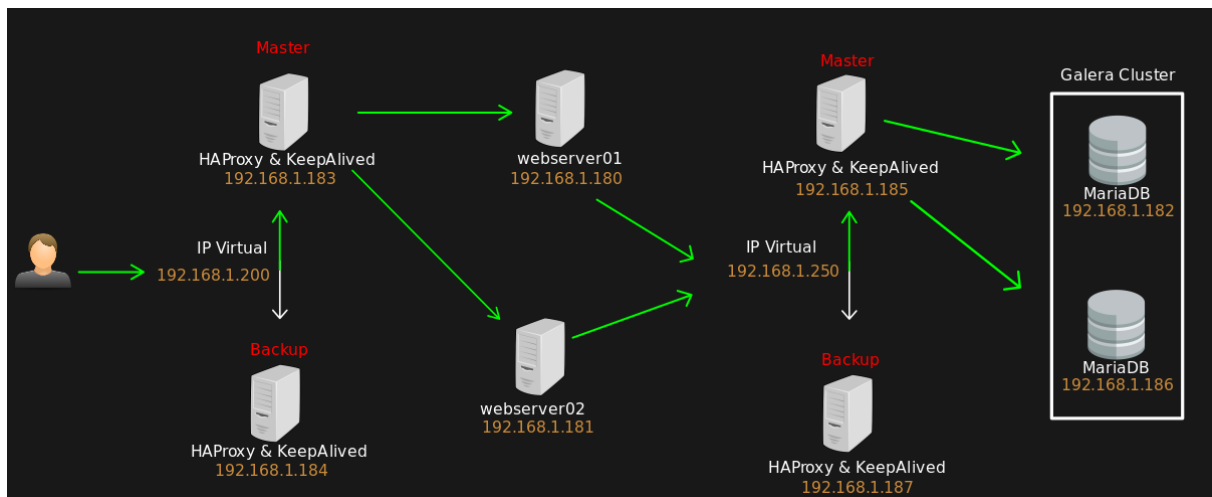


Figura 2.30: Esquema - Experiência 04

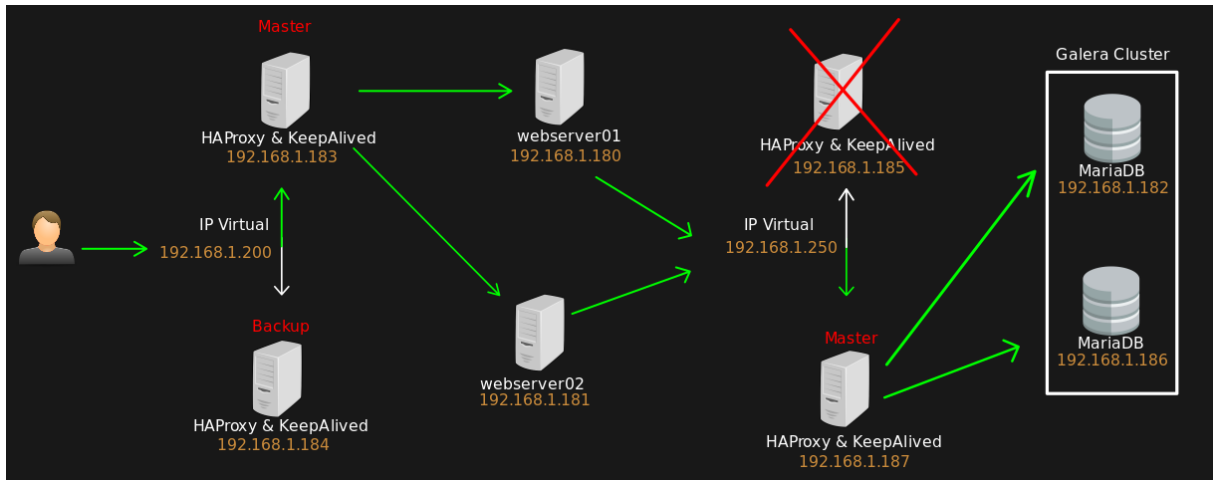


Figura 2.31: Esquema com *failover* - Experiência 04

## Resultado

Foram feitas capturas, tanto no **HAProxy03(MASTER)** como no **HAProxy04(BACKUP)** usando o *wireshark*.

Como visto anteriormente, o HAProxy03(192.168.1.185) como é o *MASTER*, envia, de segundo em segundo, um *announcement* indicando a sua prioridade enquanto que o HAProxy04(192.168.1.187) está à escuta. Na captura do *wireshark* é possível também ver o *announcement* vindo do HAProxy01(192.168.1.183).

1657 399.422756	192.168.1.185	224.0.0.18	VRRP	54 Announcement (v2)
1658 399.422938	192.168.1.183	224.0.0.18	VRRP	60 Announcement (v2)
1662 400.423248	192.168.1.185	224.0.0.18	VRRP	54 Announcement (v2)

Figura 2.32: *Announcements* do HAProxy01 e HAProxy03 - Experiência 04

Depois de desligar o serviço HAProxy do HAProxy03(192.168.1.185), o mesmo fica com uma prioridade de 91 passando assim para o estado de *BACKUP* ao mesmo tempo que o HAProxy04(192.168.1.187) passa para o estado de *MASTER* uma vez que a sua prioridade é superior (100).

```

brun0@haproxy03:~$ sudo service keepalived status
● keepalived.service - Keepalived Daemon (LVS and VRRP)
   Loaded: loaded (/lib/systemd/system/keepalived.service; enabled; vendor preset: enabled)
   Active: active (running) since Sat 2021-12-11 17:57:36 UTC; 4h 53min ago
     Main PID: 10393 (keepalived)
       Tasks: 2 (limit: 2278)
      Memory: 4.1M
     CGroup: /system.slice/keepalived.service
             └─10393 /usr/sbin/keepalived --dont-fork
               └─10404 /usr/sbin/keepalived --dont-fork

Dec 11 22:51:14 haproxy03 Keepalived_vrrp[10404]: (V1_1) Entering MASTER STATE
Dec 11 22:51:14 haproxy03 Keepalived_vrrp[10404]: (V1_1) Master received advert from 192.168.1.183
Dec 11 22:51:14 haproxy03 Keepalived_vrrp[10404]: (V1_1) Entering BACKUP STATE
Dec 11 22:51:15 haproxy03 Keepalived_vrrp[10404]: (V1_1) Ip address associated with VRID 11 not
Dec 11 22:51:16 haproxy03 Keepalived_vrrp[10404]: (V1_1) Ip address associated with VRID 11 not
Dec 11 22:51:17 haproxy03 Keepalived_vrrp[10404]: (V1_1) Ip address associated with VRID 11 not
Dec 11 22:51:18 haproxy03 Keepalived_vrrp[10404]: (V1_1) Ip address associated with VRID 11 not
Dec 11 22:51:19 haproxy03 Keepalived_vrrp[10404]: (V1_1) Ip address associated with VRID 11 not
Dec 11 22:51:20 haproxy03 Keepalived_vrrp[10404]: (V1_1) Ip address associated with VRID 11 not
Dec 11 22:51:21 haproxy03 Keepalived_vrrp[10404]: (V1_1) Ip address associated with VRID 11 not
[Lines 1-20/20 (END)]

brun0@haproxy04:~$ sudo service keepalived status
● keepalived.service - Keepalived Daemon (LVS and VRRP)
   Loaded: loaded (/lib/systemd/system/keepalived.service; enabled; vendor preset: enabled)
   Active: active (running) since Sat 2021-12-11 17:57:19 UTC; 4h 53min ago
     Main PID: 27735 (keepalived)
       Tasks: 2 (limit: 1070)
      Memory: 4.1M
     CGroup: /system.slice/keepalived.service
             └─27735 /usr/sbin/keepalived --dont-fork
               └─27745 /usr/sbin/keepalived --dont-fork

Dec 11 22:50:56 haproxy04 Keepalived_vrrp[27745]: (V1_1) Entering MASTER STATE
Dec 11 22:50:59 haproxy04 Keepalived_vrrp[27745]: (V1_1) Ip address associated with VRID 11 not
Dec 11 22:50:59 haproxy04 Keepalived_vrrp[27745]: (V1_1) Ip address associated with VRID 11 not
Dec 11 22:51:00 haproxy04 Keepalived_vrrp[27745]: (V1_1) Ip address associated with VRID 11 not
Dec 11 22:51:01 haproxy04 Keepalived_vrrp[27745]: (V1_1) Ip address associated with VRID 11 not
Dec 11 22:51:02 haproxy04 Keepalived_vrrp[27745]: (V1_1) Ip address associated with VRID 11 not
Dec 11 22:51:05 haproxy04 Keepalived_vrrp[27745]: (V1_1) Ip address associated with VRID 11 not
Dec 11 22:51:05 haproxy04 Keepalived_vrrp[27745]: (V1_1) Ip address associated with VRID 11 not
Dec 11 22:51:06 haproxy04 Keepalived_vrrp[27745]: (V1_1) Ip address associated with VRID 11 not
Dec 11 22:51:07 haproxy04 Keepalived_vrrp[27745]: (V1_1) Ip address associated with VRID 11 not
[Lines 1-20/20 (END)]

```

Figura 2.33: Troca de *MASTER* e *BACKUP* - Experiência 04

Neste momento continua a haver uma conexão normal do cliente para os *webserver*s, sendo que os mesmos continuam a conseguir fazer *queries* às bases de dados como se nada tivesse acontecido.

Depois de ser reativado o serviço de HAProxy no HAProxy03, o mesmo passa novamente a ser *MASTER* visto que a sua prioridade volta a aumentar.

```
brun0@haproxy03:~$ sudo service keepalived status
● keepalived.service - Keepalived Daemon (LVS and VRRP)
   Loaded: loaded (/lib/systemd/system/keepalived.service; enabled; vendor preset: enabled)
   Active: active (running) since Sat 2021-12-11 17:57:36 UTC; 5h 5min ago
     Main PID: 10393 (keepalived)
        Tasks: 2 (limit: 2278)
       Memory: 3.9M
      CGroup: /system.slice/keepalived.service
              └─10393 /usr/sbin/keepalived --dont-fork
                 10404 /usr/sbin/keepalived --dont-fork

Dec 11 23:02:43 haproxy03 Keepalived_vrrp[10404]: (V1_1) Changing effective priority from 91
Dec 11 23:02:43 haproxy03 Keepalived_vrrp[10404]: (V1_1) Ip address associated with VRID 11
Dec 11 23:02:43 haproxy03 Keepalived_vrrp[10404]: (V1_1) received lower priority (100) advert
Dec 11 23:02:44 haproxy03 Keepalived_vrrp[10404]: (V1_1) received lower priority (100) advert
Dec 11 23:02:44 haproxy03 Keepalived_vrrp[10404]: (V1_1) Ip address associated with VRID 11
Dec 11 23:02:45 haproxy03 Keepalived_vrrp[10404]: (V1_1) received lower priority (100) advert
Dec 11 23:02:45 haproxy03 Keepalived_vrrp[10404]: (V1_1) Ip address associated with VRID 11
Dec 11 23:02:46 haproxy03 Keepalived_vrrp[10404]: (V1_1) Entering MASTER STATE
Dec 11 23:02:49 haproxy03 Keepalived_vrrp[10404]: (V1_1) Ip address associated with VRID 11
Dec 11 23:02:49 haproxy03 Keepalived_vrrp[10404]: (V1_1) Ip address associated with VRID 11

brun0@haproxy04:~$ sudo service keepalived status
● keepalived.service - Keepalived Daemon (LVS and VRRP)
   Loaded: loaded (/lib/systemd/system/keepalived.service; enabled; vendor preset: enabled)
   Active: active (running) since Sat 2021-12-11 17:57:19 UTC; 5h 5min ago
     Main PID: 27735 (keepalived)
        Tasks: 2 (limit: 1070)
       Memory: 3.9M
      CGroup: /system.slice/keepalived.service
              └─27735 /usr/sbin/keepalived --dont-fork
                 27745 /usr/sbin/keepalived --dont-fork

Dec 11 23:02:41 haproxy04 Keepalived_vrrp[27745]: (V1_1) Ip address associated with VRID 11 not
Dec 11 23:02:42 haproxy04 Keepalived_vrrp[27745]: (V1_1) Ip address associated with VRID 11 not
Dec 11 23:02:43 haproxy04 Keepalived_vrrp[27745]: (V1_1) Ip address associated with VRID 11 not
Dec 11 23:02:44 haproxy04 Keepalived_vrrp[27745]: (V1_1) Ip address associated with VRID 11 not
Dec 11 23:02:45 haproxy04 Keepalived_vrrp[27745]: (V1_1) Ip address associated with VRID 11 not
Dec 11 23:02:46 haproxy04 Keepalived_vrrp[27745]: (V1_1) Master received advert from 192.168.1.
Dec 11 23:02:49 haproxy04 Keepalived_vrrp[27745]: (V1_1) Ip address associated with VRID 11 not
Dec 11 23:02:49 haproxy04 Keepalived_vrrp[27745]: (V1_1) Ip address associated with VRID 11 not
Dec 11 23:02:50 haproxy04 Keepalived_vrrp[27745]: (V1_1) Ip address associated with VRID 11 not
```

Figura 2.34: Reativado serviço HAProxy em HAProxy03 - Experiência 04

Por fim, podemos concluir que com esta arquitetura temos uma infraestrutura minimamente segura, capaz de ser disponível e de garantir desempenho. Tudo isto se deve aos vários *proxies* implementados assim como o *galera cluster* que oferece uma redundância enorme para um bom desempenho da infraestrutura. A implementação final, não apresenta nenhum SPOF, cumprindo assim o objetivo proposto no início do trabalho.

## Capítulo 3

# Conclusão

Com a execução destas experiências foi possível perceber a importância que tem a disponibilidade e o desempenho numa infraestrutura que fornece serviços *web*.

Inicialmente existia um pequeno problema (**como resolver 2 SPOFs?**), porém com a resolução desses problemas, foram aparecendo outros que à primeira vista aparentavam ser pequenos, no entanto os mesmos prejudicavam a infraestrutura num todo, fazendo com que a mesma ficasse inoperacional.

A partir daqui foi perceptível que antes de ser implementada e tornada operacional uma infraestrutura que fornece um serviço qualquer a mesma tem de ser bem planeada de modo a não ser preciso acrescentar muitos mais recursos aos inicialmente pensados.

Para isso é necessário criar um planeamento a pensar em todos os *single point of failures* que possam existir e como os conseguimos resolver, tornando assim a infraestrutura disponível e ao mesmo tempo com um bom desempenho fazendo uso do balanceamento de carga.

# Bibliografia

- [1] Stack Overflow  
<https://stackoverflow.com/>
- [2] Mariadb e Galera Cluster  
<https://mariadb.com/kb/en/what-is-mariadb-galera-cluster/>
- [3] Flask  
<https://flask.palletsprojects.com/en/2.0.x/>
- [4] HAProxy  
<https://www.digitalocean.com/community/tutorials/an-introduction-to-haproxy-and-load-balancing-concepts>
- [5] Keepalived  
<https://www.redhat.com/sysadmin/keepalived-basics>