Práctica 3

El desarrollo de esta práctica se basa en un balanceador de carga para las dos máquinas virtuales que creamos en las prácticas anteriores. Para ello siguiendo el esquema general de la presentación crearemos una nueva máquina virtual copia directa de las anteriores y le cambiaremos la dirección IP estática. Una vez creada y configurada correctamente comenzaremos a ejecutar los comandos indicados para instalar los balanceadores de carga indicados:

- sudo apt-get update
- sudo apt-get dist-upgrade
- sudo apt-get autoremove
- **sudo apt-get install nginx**

```
laguilarg99@UbuntuServer:~$ nginx -v
nginx version: nginx/1.10.3 (Ubuntu)
```

Como se trata de una copia de las máquinas anteriores que tenían Apache Server corriendo en la 0.0.0.0:80, es decir, en el servidor local, es necesario eliminar dicho servicio de tal puerto para que de esta forma quede libre para nginx, dicho esto debemos ejecutar como administradores:

- systemctl stop apache2
- systemctl start nginx

Ahora hay que configurar nginx adecuadamente para que funcione como balanceador:

- nano /etc/nginx/nginx.conf

```
include /etc/nginx/conf.d/*.conf;
include /etc/nginx/sites-enabled/*;
```

```
include /etc/nginx/conf.d/*.conf;
#include /etc/nginx/sites-enabled/*;
```

Después hay que configurar el upstream para las máquinas a las que repartir el tráfico:

- sudo nano /etc/nginx/conf.d/default.conf

```
upstream servidoresSWAP{
        server 192.168.56.101;
        server 192.168.56.102;
}

server{
        listen 80;
        server_name balanceador;

        access_log /var/log/nginx/balanceador.access.log;
        error_log /var/log/nginx/balanceador.error.log;
        root /var/www/;

        location /
        {
                proxy_pass http://servidoresSWAP;
                proxy_set_header Host $host;
                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
                proxy_http_version 1.1;
                proxy_set_header Connection "";

        }

}
```

Si hacemos curl varias veces curl http://192.168.56.103 devuelve de forma alterna las web de la M1 y la M2. En el caso de la captura de abajo se pondera la importancia de cada servidor.

```
upstream servidoresSWAP{
        server 192.168.56.101 weight=2;
        server 192.168.56.102 weight=1;
}

server{
        listen 80;
        server_name balanceador;

        access_log /var/log/nginx/balanceador.access.log;
        error_log /var/log/nginx/balanceador.error.log;
        root /var/www/;

        location /
        {
                proxy_pass http://servidoresSWAP;
                proxy_set_header Host $host;
                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
                proxy_http_version 1.1;
                proxy_set_header Connection "";

        }

}
```

Tras haber instalado nginx instalaremos **haproxy**,

- **sudo apt-get install haproxy**

Una vez hecho esto configuraremos el archivo de haproxy (/etc/haproxy/haproxy.cfg):

- nano /etc/haproxy/haproxy.cfg

```
frontend http-in
        bind *:80
        default_backend servidoresSWAP

backend servidoresSWAP
        server          m1 192.168.56.101:80 maxconn 32
        server          m2 192.168.56.102:80 maxconn 32_
```

El funcionamiento resultante será exactamente igual que en el primer caso con nginx, Round-Robin.

Finalmente probaremos con Apache Benchmark el rendimiento de los balanceadores, en primer lugar probaremos con **haproxy**:

luis@luis-GL553VD:~$ ab -n 1000 -c 10 http://192.168.56.103/index.html
This is ApacheBench, Version 2.3 <$Revision: 1807734 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 192.168.56.103 (be patient)
Completed 100 requests
Completed 200 requests
Completed 300 requests
Completed 400 requests
Completed 500 requests
Completed 600 requests
Completed 700 requests
Completed 800 requests
Completed 900 requests
Completed 1000 requests
Finished 1000 requests

Server Software: Apache/2.4.18
Server Hostname: 192.168.56.103
Server Port: 80

Document Path: /index.html
Document Length: 11324 bytes

Concurrency Level: 10
Time taken for tests: 0.567 seconds
Complete requests: 1000
Failed requests: 0
Total transferred: 11598000 bytes
HTML transferred: 11324000 bytes
Requests per second: 1763.63 [#/sec] (mean)
Time per request: 5.670 [ms] (mean)
Time per request: 0.567 [ms] (mean, across all concurrent requests)
Transfer rate: 19975.22 [Kbytes/sec] received

Connection Times (ms)

	min	mean[+/-sd]	median	max	
Connect:	0	0	0.0	0	0
Processing:	2	6	0.9	5	10
Waiting:	2	5	0.9	5	10
Total:	3	6	0.9	6	10

WARNING: The median and mean for the processing time are not within a normal deviation
These results are probably not that reliable.

Percentage of the requests served within a certain time (ms)
50% 6
66% 6
75% 6
80% 6
90% 7
95% 7
98% 8
99% 8
100% 10 (longest request)

Ahora probaremos el mismo programa de benchmarking con **nginx:**

```
luis@luis-GL553VD:~$ ab -n 1000 -c 10 http://192.168.56.103/index.html
This is ApacheBench, Version 2.3 <$Revision: 1807734 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 192.168.56.103 (be patient)
Completed 100 requests
Completed 200 requests
Completed 300 requests
Completed 400 requests
Completed 500 requests
Completed 600 requests
Completed 700 requests
Completed 800 requests
Completed 900 requests
Completed 1000 requests
Finished 1000 requests


Server Software:        nginx/1.10.3
Server Hostname:        192.168.56.103
Server Port:            80

Document Path:          /index.html
Document Length:        11324 bytes

Concurrency Level:      10
Time taken for tests:   0.642 seconds
Complete requests:      1000
Failed requests:        0
Total transferred:      11597000 bytes
HTML transferred:       11324000 bytes
Requests per second:    1558.27 [#/sec] (mean)
Time per request:       6.417 [ms] (mean)
Time per request:       0.642 [ms] (mean, across all concurrent requests)
Transfer rate:          17647.67 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.0      0       1
Processing:     2    6   1.8      6      25
Waiting:        2    6   1.8      6      25
Total:          2    6   1.8      6      26
```

Percentage of the requests served within a certain time (ms)
  50% 6
  66% 7
  75% 7
  80% 7
  90% 7
  95% 7
  98% 8
  99% 19
 100% 26 (longest request)

Ahora probaremos el mismo programa de benchmarking con **nginx ponderado dándole prioridad a la primera máquina:**

luis@luis-GL553VD:~$ ab -n 1000 -c 10 http://192.168.56.103/index.html
This is ApacheBench, Version 2.3 <$Revision: 1807734 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 192.168.56.103 (be patient)
Completed 100 requests
Completed 200 requests
Completed 300 requests
Completed 400 requests
Completed 500 requests
Completed 600 requests
Completed 700 requests
Completed 800 requests
Completed 900 requests
Completed 1000 requests
Finished 1000 requests

Server Software: nginx/1.10.3
Server Hostname: 192.168.56.103
Server Port: 80

Document Path: /index.html
Document Length: 11324 bytes

Concurrency Level: 10
Time taken for tests: 0.622 seconds
Complete requests: 1000
Failed requests: 0
Total transferred: 11597000 bytes
HTML transferred: 11324000 bytes

```
Requests per second:    1606.70 [#/sec] (mean)
Time per request:       6.224 [ms] (mean)
Time per request:       0.622 [ms] (mean, across all concurrent requests)
Transfer rate:          18196.21 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.0      0       0
Processing:     2    6   0.8      6       9
Waiting:        2    6   0.7      6       9
Total:          3    6   0.8      6       9

Percentage of the requests served within a certain time (ms)
  50%      6
  66%      6
  75%      7
  80%      7
  90%      7
  95%      7
  98%      8
  99%      9
 100%      9 (longest request)
```

Tras ejecutar el benchmark en las tres posibles configuraciones, observamos que es nginx con la primera máquina con mayor peso la que mejores prestaciones consigue.