Práctica 3: Balanceo de carga en un sitio web

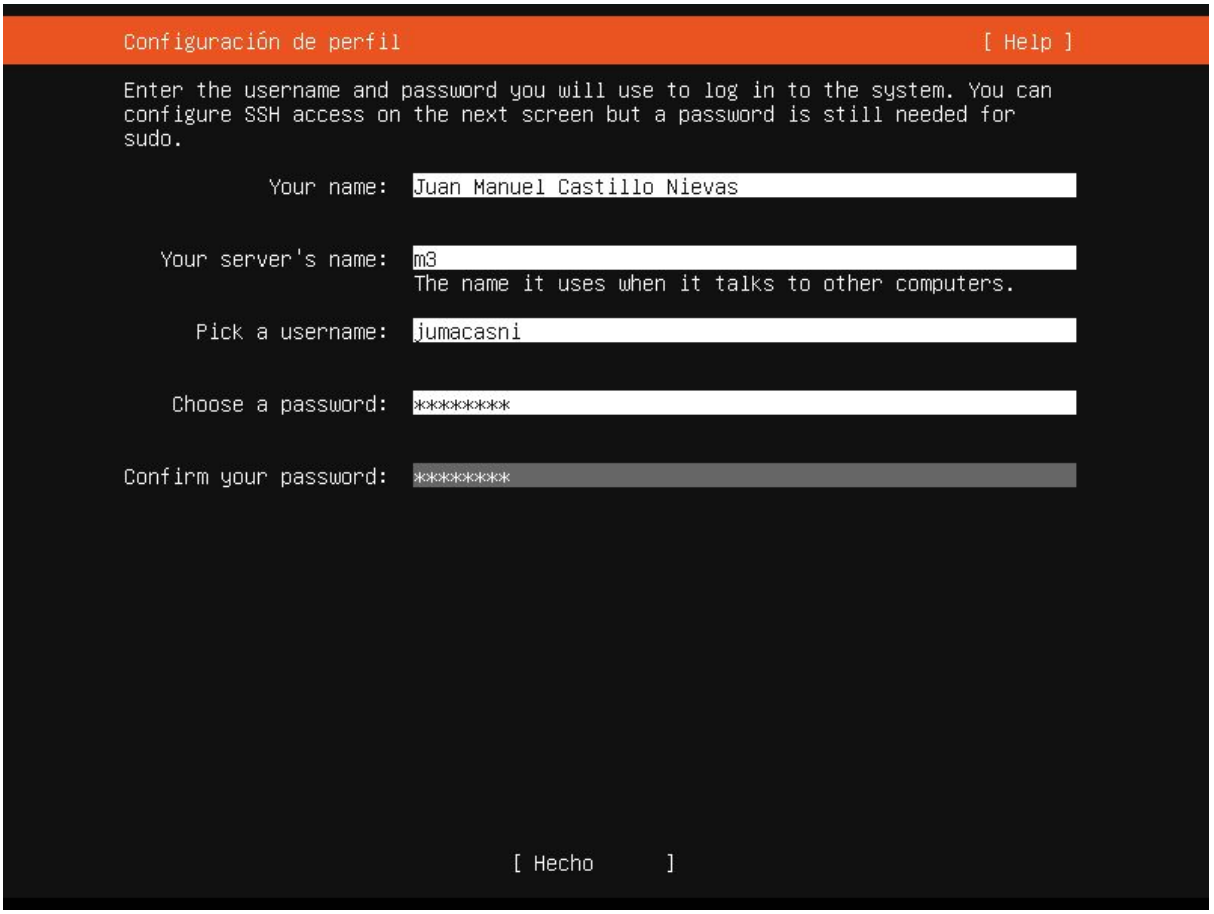
Juan Manuel Castillo Nievas

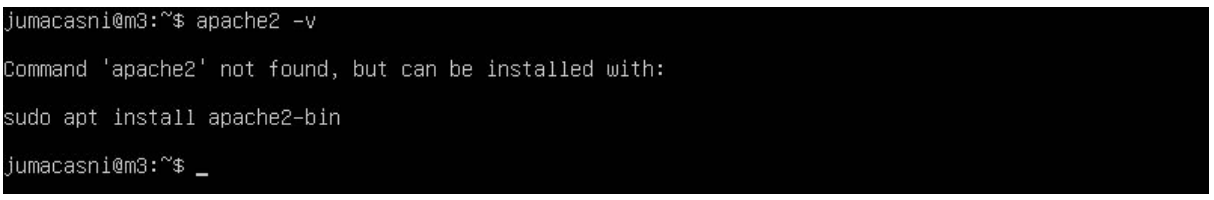
A partir de esta práctica, la IP de m1 es 192.168.56.102 y la IP de m2 es 192.168.56.103. He tenido que volver a realizar la práctica 1 porque configuré mal el netplan y no tenía acceso a Internet, por lo que decidí volver a instalar m1 y m2 de forma correcta.

1. Configurar una máquina e instalar el nginx como balanceador de carga

---

Para empezar he instalado una nueva máquina m3 y me he asegurado de que no tiene apache instalado.





A continuación, he instalado nginx mediante la línea de comandos y me he asegurado de que está en activo.

```
jumacasni@m3:~$ sudo systemctl start nginx
jumacasni@m3:~$ sudo service nginx status
● nginx.service - A high performance web server and a reverse proxy server
   Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: enabled)
   Active: active (running) since Tue 2020-03-31 15:30:25 UTC; 16s ago
     Docs: man:nginx(8)
 Main PID: 13636 (nginx)
    Tasks: 2 (limit: 2318)
   CGroup: /system.slice/nginx.service
           ├─13636 nginx: master process /usr/sbin/nginx -g daemon on; master_process on;
           └─13639 nginx: worker process

Mar 31 15:30:25 m3 systemd[1]: Starting A high performance web server and a reverse proxy server...
Mar 31 15:30:25 m3 systemd[1]: nginx.service: Failed to parse PID from file /run/nginx.pid: Invalid
Mar 31 15:30:25 m3 systemd[1]: Started A high performance web server and a reverse proxy server.
lines 1-13/13 (END)
```

Ahora hay que configurar nginx para redirigir el tráfico a un grupo de servidores, en nuestro caso las máquinas m1 y m2.

En m3 he creado el archivo *default.conf* con la información correspondiente y he reiniciado *nginx*.

```
jumacasni@m3:~$ cat /etc/nginx/conf.d/default.conf
upstream servidoresSWAP{
        server 192.168.56.102;
        server 192.168.56.103;
}

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 $proxy_add_x_forwarded_for;
                proxy_http_version 1.1;
                proxy_set_header Connection "";
        }
}
jumacasni@m3:~$ sudo service nginx restart
jumacasni@m3:~$ _
```

Ahora voy a mostrar la IP de m3 para saber a qué dirección tengo que hacer curl:

```
jumacasni@m3:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::a00:27ff:fe43:5806  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:43:58:06  txqueuelen 1000  (Ethernet)
        RX packets 7917  bytes 9746125 (9.7 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1562  bytes 115686 (115.6 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

enp0s8: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.56.105  netmask 255.255.255.0  broadcast 192.168.56.255
        inet6 fe80::a00:27ff:fe3f:7f5b  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:3f:7f:5b  txqueuelen 1000  (Ethernet)
        RX packets 177  bytes 24670 (24.6 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 166  bytes 23747 (23.7 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 214  bytes 17170 (17.1 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 214  bytes 17170 (17.1 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

jumacasni@m3:~$ _
```

También he modificado el archivo *index.html* de cada máquina m1 y m2 para ver la diferencia de cada máquina. A continuación, desde una nueva máquina virtual m4 he hecho curl hacia la IP de m3 y como resultado he obtenido el html de m1 y m2 de forma alternativa:

```
jumacasni@m4:~$ curl http://192.168.56.105/
<HTML>
<BODY>
Estás viendo M1!
Práctica 3 realizada por Juan Manuel Castillo Nievas, SWAP 2019/2020
</BODY>
</HTML>
jumacasni@m4:~$ curl http://192.168.56.105/
<HTML>
<BODY>
Estás viendo M2!
Práctica 3 realizada por Juan Manuel Castillo Nievas, SWAP 2019/2020
</BODY>
</HTML>
jumacasni@m4:~$ curl http://192.168.56.105/
<HTML>
<BODY>
Estás viendo M1!
Práctica 3 realizada por Juan Manuel Castillo Nievas, SWAP 2019/2020
</BODY>
</HTML>
jumacasni@m4:~$ curl http://192.168.56.105/
<HTML>
<BODY>
Estás viendo M2!
Práctica 3 realizada por Juan Manuel Castillo Nievas, SWAP 2019/2020
</BODY>
</HTML>
jumacasni@m4:~$
```

El siguiente paso es configurar el balanceo que se quiera. Por ejemplo, yo he utilizado el parámetro *weight* para que, de cada 3 peticiones, 2 vayan hacia m2 y 1 vaya hacia m1:

```
jumacasni@m3:~$ cat /etc/nginx/conf.d/default.conf
upstream servidoresSWAP{
        server 192.168.56.102 weight=1;
        server 192.168.56.103 weight=2;
}
```

Y, efectivamente, el resultado es el siguiente:

```
jumacasni@m4:~$ curl http://192.168.56.105/
<HTML>
<BODY>
Estás viendo M1!
Práctica 3 realizada por Juan Manuel Castillo Nievas, SWAP 2019/2020
</BODY>
</HTML>
jumacasni@m4:~$ curl http://192.168.56.105/
<HTML>
<BODY>
Estás viendo M2!
Práctica 3 realizada por Juan Manuel Castillo Nievas, SWAP 2019/2020
</BODY>
</HTML>
jumacasni@m4:~$ curl http://192.168.56.105/
<HTML>
<BODY>
Estás viendo M2!
Práctica 3 realizada por Juan Manuel Castillo Nievas, SWAP 2019/2020
</BODY>
</HTML>
jumacasni@m4:~$ curl http://192.168.56.105/
<HTML>
<BODY>
Estás viendo M1!
Práctica 3 realizada por Juan Manuel Castillo Nievas, SWAP 2019/2020
</BODY>
</HTML>
jumacasni@m4:~$ _
```

También se puede hacer un balanceo por IP utilizando la directiva *ip_hash*.
Además, en *nginx* ya se puede utilizar *keepalive* para realizar una conexión con una persistencia de múltiples peticiones HTTP.

2. Configurar una máquina e instalar el haproxy como balanceador de carga

Primero he parado el servicio de *nginx* en m3. A continuación, he instalado *haproxy* y he modificado el archivo de configuración añadiendo las siguientes líneas:

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

backend servidoresSWAP
        server  m1 192.168.56.102:80 maxconn 32
        server  m2 192.168.56.103:80 maxconn 32
jumacasni@m3:~$ _
```

Y he reiniciado el servicio de haproxy con *sudo service haproxy restart*. Desde m4, he hecho curl hacia m3 y he obtenido el mismo resultado que con *nginx*:

```
jumacasni@m4:~$ curl http://192.168.56.105/
<HTML>
<BODY>
Estás viendo M1!
Práctica 3 realizada por Juan Manuel Castillo Nievas, SWAP 2019/2020
</BODY>
</HTML>
jumacasni@m4:~$ curl http://192.168.56.105/
<HTML>
<BODY>
Estás viendo M2!
Práctica 3 realizada por Juan Manuel Castillo Nievas, SWAP 2019/2020
</BODY>
</HTML>
jumacasni@m4:~$ _
```

3. Someter a la granja web a una alta carga, generada con la herramienta Apache Benchmark, teniendo primero nginx y después haproxy.

Primero he parado haproxy y he puesto nginx en marcha. He ejecutado *ab -n 10000-c 10 http://ip_maquinaM3/index.html* y he obtenido los siguientes resultados:

```
Server Software:        nginx/1.14.0
Server Hostname:        192.168.56.105
Server Port:            80

Document Path:          /index.html
Document Length:        118 bytes

Concurrency Level:      10
Time taken for tests:   8.059 seconds
Complete requests:      10000
Failed requests:        0
Total transferred:      3870000 bytes
HTML transferred:       1180000 bytes
Requests per second:    1240.90 [#/sec] (mean)
Time per request:       8.059 [ms] (mean)
Time per request:       0.806 [ms] (mean, across all concurrent requests)
Transfer rate:          468.97 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.2      0       9
Processing:     2    8   0.6      8      18
Waiting:        0    7   0.6      7      16
Total:          4    8   0.6      8      18

Percentage of the requests served within a certain time (ms)
  50%      8
  66%      8
  75%      8
  80%      8
  90%      8
  95%      9
  98%      9
  99%      9
 100%     18 (longest request)
jumacasni@m4:~$ _
```

Con haproxy en marcha he vuelto a ejecutar lo mismo y he obtenido lo siguiente:

```
Server Software:        Apache/2.4.29
Server Hostname:        192.168.56.105
Server Port:            80

Document Path:          /index.html
Document Length:        118 bytes

Concurrency Level:      10
Time taken for tests:   6.707 seconds
Complete requests:      10000
Failed requests:        0
Total transferred:      3880000 bytes
HTML transferred:       1180000 bytes
Requests per second:    1490.91 [#/sec] (mean)
Time per request:       6.707 [ms] (mean)
Time per request:       0.671 [ms] (mean, across all concurrent requests)
Transfer rate:          564.91 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.2      0       3
Processing:     4    6   0.6      6      15
Waiting:        4    6   0.5      6      15
Total:          4    7   0.6      7      16

Percentage of the requests served within a certain time (ms)
  50%      7
  66%      7
  75%      7
  80%      7
  90%      7
  95%      8
  98%      8
  99%      8
 100%     16 (longest request)
jumacasni@m4:~$
```

Se puede apreciar que haproxy obtiene mejores resultados que nginx. La media total de los tiempos de conexión en haproxy es 7ms y en nginx es de 8ms.

4. OPCIONAL: uso de Varnish como balanceador

Como ejercicio opcional, he instalado Varnish en m3 usando *sudo apt-get install varnish* y he mirado en su página la documentación para hacer un round-robin en el balanceador.

https://varnish-cache.org/docs/4.1/users-guide/vcl-backends.html
En la sección de *Directors* viene lo que hay que poner en el archivo de configuración para un round-robin.

Entonces he modificado el archivo */etc/varnish/default.vcl* con la siguiente información:

```
# Default backend definition. Set this to point to your content server.
backend m1 {
    .host = "192.168.56.102";
}

backend m2 {
    .host = "192.168.56.103";
}

sub vcl_init {
    new bar = directors.round_robin();
    bar.add_backend(m1);
    bar.add_backend(m2);
}

sub vcl_recv {
    set req.backend_hint = bar.backend();
}

sub vcl_backend_response {
    # Happens after we have read the response headers from the backend.
    #
    # Here you clean the response headers, removing silly Set-Cookie headers
    # and other mistakes your backend does.
}

sub vcl_deliver {
    # Happens when we have all the pieces we need, and are about to send the
    # response to the client.
    #
    # You can do accounting or modifying the final object here.
}


jumacasni@m3:~$ sudo systemctl restart varnish
jumacasni@m3:~$ _
```

Y he reiniciado el servicio.

A continuación he probado a hacer curl desde m4 hacia m3 pero por alguna razón solo obtengo la información de m1. He buscado información para ver a qué se puede deber y en StackOverflow he encontrado a alguien que le pasaba algo parecido y que tiene que ver con algún error de módulo de varnish:
https://stackoverflow.com/questions/36852108/varnish-round-robin-director-not-picking-backends

```
jumacasni@m4:~$ curl http://192.168.56.105/
<HTML>
<BODY>
Estás viendo M1!
Práctica 3 realizada por Juan Manuel Castillo Nievas, SWAP 2019/2020
</BODY>
</HTML>
jumacasni@m4:~$ curl http://192.168.56.105/
<HTML>
<BODY>
Estás viendo M1!
Práctica 3 realizada por Juan Manuel Castillo Nievas, SWAP 2019/2020
</BODY>
</HTML>
jumacasni@m4:~$ curl http://192.168.56.105/
<HTML>
<BODY>
Estás viendo M1!
Práctica 3 realizada por Juan Manuel Castillo Nievas, SWAP 2019/2020
</BODY>
</HTML>
jumacasni@m4:~$ curl http://192.168.56.105/
<HTML>
<BODY>
Estás viendo M1!
Práctica 3 realizada por Juan Manuel Castillo Nievas, SWAP 2019/2020
</BODY>
</HTML>
```

A pesar de ello, he utilizado el comando de *ab* para someter una gran carga y he obtenido lo siguiente:

```
Server Software:        Apache/2.4.29
Server Hostname:        192.168.56.105
Server Port:            80

Document Path:          /index.html
Document Length:        118 bytes

Concurrency Level:      10
Time taken for tests:   6.422 seconds
Complete requests:      10000
Failed requests:        0
Total transferred:      4596308 bytes
HTML transferred:       1180000 bytes
Requests per second:    1557.21 [#/sec] (mean)
Time per request:       6.422 [ms] (mean)
Time per request:       0.642 [ms] (mean, across all concurrent requests)
Transfer rate:          698.97 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.3      0      13
Processing:     0    6   4.6      6      98
Waiting:        0    5   4.2      5      97
Total:          1    6   4.6      6      98

Percentage of the requests served within a certain time (ms)
  50%      6
  66%      8
  75%      9
  80%     10
  90%     12
  95%     14
  98%     17
  99%     20
 100%     98 (longest request)
jumacasni@m4:~$
```