

SWAP: Balanceo de carga en un sitio web

David Cabezas Berrido

dxabezas@correo.ugr.es

12 de abril de 2021

Índice

1. Preparativos	2
2. Balanceo de carga con NGINX	2
3. Balanceo de carga con HAproxy	4

1. Preparativos

Creamos dos archivos `/var/www/html/index.html` básicos en las máquinas 1 y 2, donde se referencia el número de la máquina a la que pertenece el archivo para saber cuál de las dos máquinas atendió la petición.

Creamos una nueva máquina virtual M3 con Ubuntu Server, pero no instalamos los servicios de la práctica 1, ya que no podemos tener a Apache ocupando el puerto 80. Nos limitamos a configurar el doble adaptador de red como hicimos en las otras dos máquinas. Su dirección IP es 192.168.56.103, para hacer peticiones desde la máquina anfitriona.

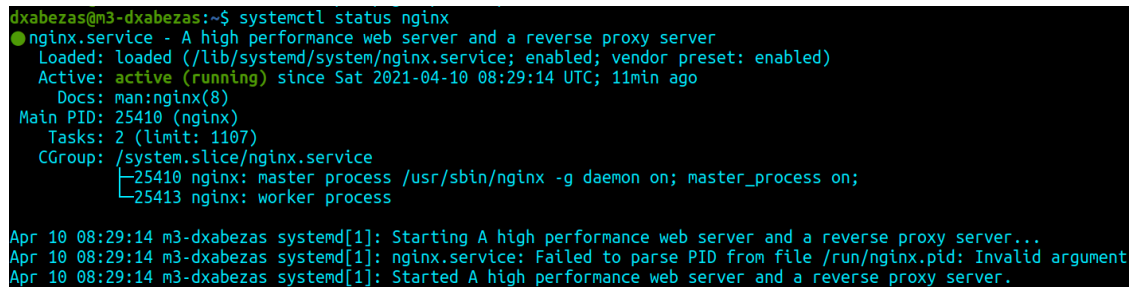
A partir de aquí, todas las órdenes y configuraciones se realizan en M3 a menos que digamos lo contrario.

2. Balanceo de carga con NGINX

Comenzamos instalando y lanzando NGINX:

```
sudo apt-get update && sudo apt-get dist-upgrade && sudo apt-get autoremove
sudo apt-get install nginx
sudo systemctl start nginx
```

Comprobamos que el servicio está en funcionamiento.

A terminal window showing the status of the nginx service. The prompt is 'dxabezas@m3-dxabezas:~\$'. The command 'systemctl status nginx' is entered. The output shows that the service is loaded and active. Below this, the logs for the service are displayed, showing the start of the service and a warning about a failed attempt to parse the PID from the file /run/nginx.pid.

```
dxabezas@m3-dxabezas:~$ systemctl status nginx
● 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 Sat 2021-04-10 08:29:14 UTC; 11min ago
     Docs: man:nginx(8)
    Main PID: 25410 (nginx)
      Tasks: 2 (limit: 1107)
   CGroup: /system.slice/nginx.service
           └─25410 nginx: master process /usr/sbin/nginx -g daemon on; master_process on;
             └─25413 nginx: worker process

Apr 10 08:29:14 m3-dxabezas systemd[1]: Starting A high performance web server and a reverse proxy server...
Apr 10 08:29:14 m3-dxabezas systemd[1]: nginx.service: Failed to parse PID from file /run/nginx.pid: Invalid argument
Apr 10 08:29:14 m3-dxabezas systemd[1]: Started A high performance web server and a reverse proxy server.
```

Figura 1: NGINX está activo.

Escribimos en `/etc/nginx/conf.d/default.conf` la configuración que se indica en el guión para que NGINX funcione como balanceador de carga en lugar de como servidor web.

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

server{
    listen 80;
    server_name balanceador_dxabezas;
    access_log /var/log/nginx/balanceador_dxabezas.access.log;
    error_log /var/log/nginx/balanceador_dxabezas.error.log;
    root /var/www/;

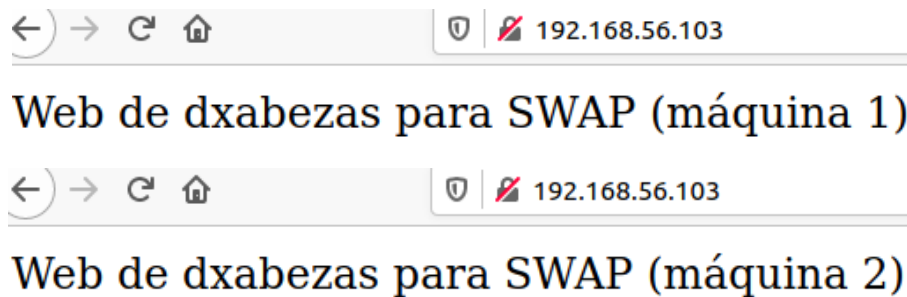
    location /
    {
        proxy_pass http://balanceo_dxabezas;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_http_version 1.1;
        proxy_set_header Connection "";
    }
}
```

Reestauramos el servicio con `sudo service nginx restart`, haremos esto (aunque no lo digamos) cada vez que modifiquemos este fichero. Ahora si visitamos la IP de M3 en el navegador de la máquina anfitriona, observamos que NGINX sigue funcionando como servidor web.



Figura 2: NGINX está funcionando como servidor web.

Como se nos indica en el guión, comentamos la línea `/etc/nginx/sites-enabled/*;` del fichero `/etc/nginx/nginx.conf`. Ahora accedemos a la IP de la máquina 3, y cada vez que refrestamos la página se turnan las máquinas 1 y 2 para servirnos su `index.html`.



Ahora añadimos el parámetro `weight` en el fichero `/etc/nginx/conf.d/default.conf` para que la máquina 2 reciba el doble de peticiones que la 1.

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

Si ahora vamos refrescando la página, la máquina que nos atiende en cada momento es: M1, M2, M2, M1, M2, M2, M1, M2, M2, ...

Seguidamente, probamos a cambiar el algoritmo de Round-Robin (por defecto) a IP-HASH para que siempre nos atienda la misma máquina, lo conseguimos añadiendo la directiva `ip_hash` en el fichero de configuración.

```
upstream balanceo_dxabezas {
    ip_hash;
    server 192.168.56.101;
    server 192.168.56.102;
}
```

Ahora, por más que refresquemos la página, nos atiende siempre la misma máquina, en nuestro caso la 2.

Finalmente, activamos las conexiones con `keepalive` the 3 segundos.

```
upstream balanceo_dxabezas {
    server 192.168.56.101;
    server 192.168.56.102;
```

```
    keepalive 3;
}
```

Aunque no es sencillo comprobar que funciona correctamente, ya que recargar la página abre una conexión nueva. Probamos algunas opciones avanzadas más. De las propuestas en el guión, elegiremos aquellas cuyo funcionamiento podamos comprobar más fácilmente.

```
upstream balanceo_dxabezas {
    ip_hash;
    server 192.168.56.101;
    server 192.168.56.102 down;
}
```

Marcando M2 como down con `ip_hash`, comprobamos que ahora nos atiende M1 en lugar de M2.

```
upstream balanceo_dxabezas {
    server 192.168.56.101;
    server 192.168.56.102 backup;
}
```

Ahora marcamos M2 como backup y siempre nos atiende M1. Si desactivamos el servicio de M1 con `sudo systemctl stop apache2`, pasa a atendernos todo el rato M2.

3. Balanceo de carga con HAProxy

Primero apagamos NGINX para que libere el puerto 80 con `sudo systemctl stop nginx`

Instalamos el balanceador con `sudo apt install haproxy`. Añadimos la siguiente configuración al fichero `/etc/haproxy/haproxy.cfg`, y hacemos `sudo systemctl restart haproxy.service` (lo haremos tras cada modificación del fichero de configuración).

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

backend balanceo_dxabezas
    server m1 192.168.56.101:80 maxconn 32
    server m2 192.168.56.102:80 maxconn 32
```

Si ahora hacemos `status`, vemos que el servicio está activo y nuestro balanceador iniciado.

```
dxabezas@m3-dxabezas:~$ sudo systemctl status haproxy
● haproxy.service - HAProxy Load Balancer
   Loaded: loaded (/lib/systemd/system/haproxy.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2021-04-12 18:11:06 UTC; 1min 32s ago
     Docs: man:haproxy(1)
           file:/usr/share/doc/haproxy/configuration.txt.gz
   Process: 3366 ExecStartPre=/usr/sbin/haproxy -f $CONFIG -c -q $EXTRA_OPTS (code=exited, status=0/SUCCESS)
  Main PID: 3376 (haproxy)
    Tasks: 2 (limit: 1107)
   CGroup: /system.slice/haproxy.service
           └─3376 /usr/sbin/haproxy -Ws -f /etc/haproxy/haproxy.cfg -p /run/haproxy.pid
             └─3377 /usr/sbin/haproxy -Ws -f /etc/haproxy/haproxy.cfg -p /run/haproxy.pid

Apr 12 18:11:06 m3-dxabezas systemd[1]: Starting HAProxy Load Balancer...
Apr 12 18:11:06 m3-dxabezas haproxy[3376]: Proxy http-in started.
Apr 12 18:11:06 m3-dxabezas haproxy[3376]: Proxy http-in started.
Apr 12 18:11:06 m3-dxabezas haproxy[3376]: Proxy balanceo_dxabezas started.
Apr 12 18:11:06 m3-dxabezas haproxy[3376]: Proxy balanceo_dxabezas started.
Apr 12 18:11:06 m3-dxabezas systemd[1]: Started HAProxy Load Balancer.
```

Si ahora accedemos a `192.168.56.103` desde el navegador, observamos que las dos máquinas se turnan para servirnos su `index.html`.

Ahora configuramos la ponderación.

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

```
backend balanceo_dxabezas
    server m1 192.168.56.101:80 maxconn 32 weight 2
    server m2 192.168.56.102:80 maxconn 32 weight 1
```

Si refrescamos la página nos atiende M1, M1, M2, M1, M1, M2, M1, M1, M2, ...

Ahora buscaremos algunas opciones avanzadas. Seleccionamos balanceo por IP-HASH.

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

backend balanceo_dxabezas
    balance source
    hash-type consistent
    server m1 192.168.56.101:80 maxconn 32
    server m2 192.168.56.102:80 maxconn 32
```

Ahora nos atiende siempre M1.

Finalmente, ponemos M1 en modo backup. Debemos añadir `check`, ya que no se activa el servidor de backup si `check` no devuelve DOWN. Si no añadimos la comprobación, las peticiones no son atendidas.

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

backend balanceo_dxabezas
    server m1 192.168.56.101:80 maxconn 32 check backup
    server m2 192.168.56.102:80 maxconn 32 check
```

Nos atiende siempre M2, pero cuando apagamos el servicio Apache2 en M2, pasa a atendernos siempre M1.

4. Estadísticas de HAproxy