



DESPLIEGUE DE APLICACIONES WEB

# BALANCEO DE CARGA CON PROXY INVERSO EN NGINX

# ÍNDICE

<b>INTRODUCCIÓN .....</b>	<b>3</b>
A. ¿QUÉ ES UN BALANCEADOR DE CARGA? .....	3
B. REQUISITOS PARA REALIZAR LA PRÁCTICA .....	4
<b>CONFIGURACIÓN PREVIA DEL SERVIDOR WEB 1 .....</b>	<b>5</b>
<b>CONFIGURACIÓN PREVIA DEL SERVIDOR WEB 2 .....</b>	<b>7</b>
<b>CONFIGURACIÓN DEL PROXY INVERSO Y BALANCEADOR DE CARGA .....</b>	<b>10</b>
<b>COMPROBACIÓN DEL FUNCIONAMIENTO DEL BALANCEO DE CARGA .....</b>	<b>12</b>
A. AMBOS SERVIDORES OPERATIVOS .....	12
B. SERVIDOR WEB 1 NO OPERATIVO .....	14
C. SERVIDOR WEB 2 NO OPERATIVO .....	15
<b>CUESTIONES FINALES .....</b>	<b>17</b>

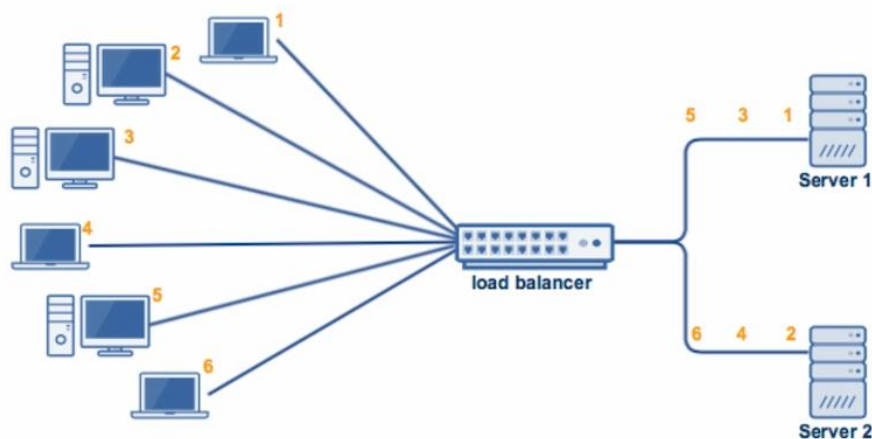
# INTRODUCCIÓN

## A. ¿QUÉ ES UN BALANCEADOR DE CARGA?

Al igual que en la anterior práctica vamos a realizar una pequeña introducción explicando los conceptos principales que se trataran en esta práctica que se basa en el uso de un proxy inverso como balanceador de carga.

Como vimos en la anterior práctica un proxy inverso es aquel tipo de proxy que se va a colocar delante de los servidores y va a aceptar las solicitudes de los clientes, reenviándolas al servidor web y devolviendo las respuestas de este hacia los clientes.

Por su parte un balanceador de carga se va a encargar de repartir las múltiples solicitudes entrantes del cliente entre un grupo de servidores y, a continuación, devolver la respuesta correspondiente a cada usuario.



Aunque ambos puedan parecer que son iguales realmente no lo son, por ejemplo, implementar un balanceador de carga sólo tendrá sentido cuando tengamos varios servidores, sin embargo, la implementación de un proxy inverso tendrá sentido incluso cuando sólo se cuenten con un servidor.

Podríamos pensar que el proxy inverso es como la cara visible que tiene nuestro sitio web, es donde está la dirección del dominio y donde van a llegar las solicitudes de los clientes encargándose en ese momento de decidir quién puede o no pasar a los servidores y ejerciendo de intermediario entre cliente y servidor impidiendo, por tanto, que el cliente acceda directamente al servidor.

La finalidad principal de tener varios servidores para un sitio web es evitar que en caso de fallo del servidor el sitio web tenga una caída completa.

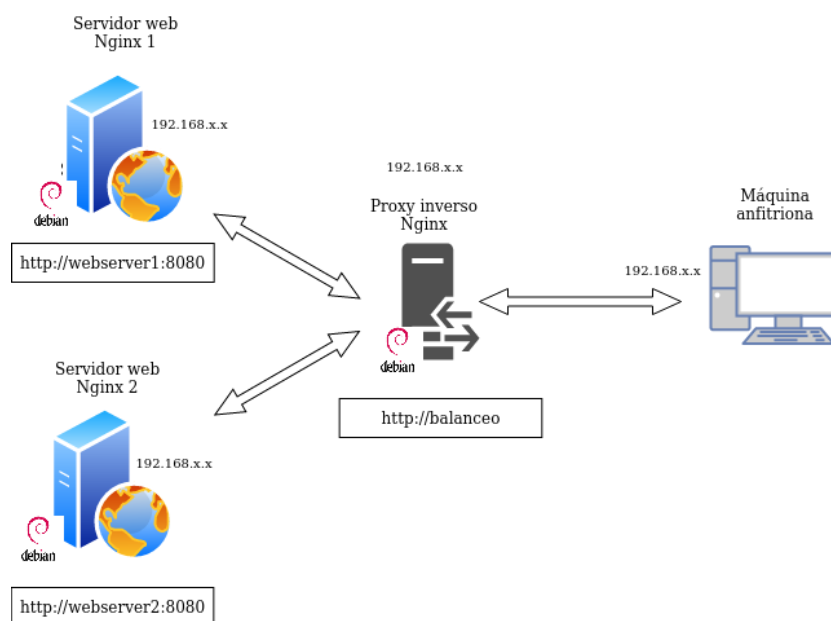
Por lo general, todos los servidores alojarán el mismo contenido y el balanceador se encargará de decidir a que servidor le corresponde cada petición que le llegue, impidiendo no sólo la sobrecarga de servidores sino el mejorar la experiencia de usuario al reducir la cantidad de respuestas de error, algo que consigue detectando en qué momento se va a producir una caída de un servidor y desviando las solicitudes de este a otros servidores.

## B. REQUISITOS PARA REALIZAR LA PRÁCTICA

En esta práctica vamos a configurar dos servidores web Nginx instalados en dos máquinas con SO Debian 12, además reutilizaremos la máquina virtual que nos sirvió como proxy inverso en la anterior práctica convirtiéndola además en un balanceador de carga.



Para comprobar el funcionamiento de nuestro sistema multi-servidor realizaremos peticiones HTTP desde nuestra máquina cliente de Ubuntu.



## CONFIGURACIÓN PREVIA DEL SERVIDOR WEB 1

En esta práctica ya no vamos a utilizar ninguno de los sitios web que hemos utilizado en prácticas anteriores, por tanto, las máquinas servidoras vamos a eliminar las referencias al sitio web webserver (que contenía los contenidos de [www.estatica.es](http://www.estatica.es)) que teníamos configurado y creando un

Comenzaremos con el directorio en `/var/www/`:

```
albertom-servidor@servidor-debian:~$ ls -la /var/www
total 12
drwxr-xr-x  3 root    root    4096 oct 27 16:09 .
drwxr-xr-x 12 root    root    4096 oct  6 19:07 ..
drwxr-xr-x  6 www-data www-data 4096 oct  9 16:52 webserver
```

Eliminaremos el directorio `/webserver/` con el comando:

```
sudo rm -Rf /var/www/webserver/
```

```
albertom-servidor@servidor-debian:~$ sudo rm -Rf /var/www/webserver
albertom-servidor@servidor-debian:~$ sudo ls -la /var/www
total 8
drwxr-xr-x  2 root    root    4096 oct 28 21:40 .
drwxr-xr-x 12 root    root    4096 oct  6 19:07 ..
```

Este servidor va a contener un sitio web estático que nos informe del número de servidor en el que nos encontramos. Para ello vamos a crear un directorio que se llamara `webserver1`:

```
sudo mkdir /var/www/webserver1/
```

```
albertom-servidor@servidor-debian:~$ sudo mkdir /var/www/webserver1/
albertom-servidor@servidor-debian:~$ ls -la /var/www/
total 12
drwxr-xr-x  3 root    root    4096 oct 29 07:03 .
drwxr-xr-x 12 root    root    4096 oct  6 19:07 ..
drwxr-xr-x  2 root    root    4096 oct 29 07:03 webserver1
```

Y en su interior tendremos un archivo con el nombre de `index.html` que funcionará como página raíz del sitio web:

```
sudo nano /var/www/webserver1/index.html
```

```
albertom-servidor@servidor-debian:~$ sudo nano /var/www/webserver1/index.html
```

Dentro del `.html` crearemos una estructura HTML por ejemplo la siguiente:

```
GNU nano 7.2 /var/www/webserver1/html/index.html
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Prueba de balanceo de carga con Nginx</title>
</head>
<body>
  <h2>Este es el servidor web 1</h2>
  <p>Comprueba el balanceo de carga con Nginx recargando esta página.</p>
</body>
</html>
```

Guardamos el archivo y vamos a darle la propiedad y los permisos pertinentes sobre el directorio /webserver1/ y el archivo que contiene al usuario del servidor web www-data para que pueda acceder sin problemas y pueda manejar estos ficheros:

```
sudo chown -R www-data:www-data /var/www/webserver1/
```

```
sudo chmod -R 755 /var/www/webserver1/
```

```
albertom-servidor@servidor-debian:~$ sudo chown -R www-data:www-data /var/www/webserver1/
albertom-servidor@servidor-debian:~$ sudo chmod -R 755 /var/www/webserver1/
albertom-servidor@servidor-debian:~$ ls -la /var/www/
total 12
drwxr-xr-x 3 root root 4096 oct 29 07:03 .
drwxr-xr-x 12 root root 4096 oct 6 19:07 ..
drwxr-xr-x 2 www-data www-data 4096 oct 29 07:08 webserver1
albertom-servidor@servidor-debian:~$ ls -la /var/www/webserver1/
total 12
drwxr-xr-x 2 www-data www-data 4096 oct 29 07:08 .
drwxr-xr-x 3 root root 4096 oct 29 07:03 ..
-rwxr-xr-x 1 www-data www-data 346 oct 29 07:08 index.html
```

Ahora pasaremos a configurar el sitio web en el directorio /etc/nginx y lo primero que debemos hacer es el unlink de webserver en el directorio /sites-enabled/

```
albertom-servidor@servidor-debian:~$ ls -la /etc/nginx/sites-enabled/
total 8
drwxr-xr-x 2 root root 4096 oct 27 16:30 .
drwxr-xr-x 8 root root 4096 oct 23 17:23 ..
lrwxrwxrwx 1 root root 36 oct 27 16:30 webserver -> /etc/nginx/sites-available/webserver
```

Para conseguirlo usaremos el comando unlink:

```
sudo unlink /etc/nginx/sites-enabled/webserver
```

```
albertom-servidor@servidor-debian:~$ sudo unlink /etc/nginx/sites-enabled/webserver
albertom-servidor@servidor-debian:~$ ls -la /etc/nginx/sites-enabled/
total 8
drwxr-xr-x 2 root root 4096 oct 28 21:53 .
drwxr-xr-x 8 root root 4096 oct 23 17:23 ..
```

A continuación, para no tener que reescribir todo el código del archivo del directorio /sites-available vamos a realizar una modificación del nombre, de webserver a webserver1:

```
sudo mv /etc/nginx/sites-available/webserver /etc/nginx/sites-available/webserver1
```

```
albertom-servidor@servidor-debian:~$ sudo mv /etc/nginx/sites-available/webserver /etc/nginx/sites-available/webserver1
albertom-servidor@servidor-debian:~$ ls -la /etc/nginx/sites-available/
total 12
drwxr-xr-x 2 root root 4096 oct 28 21:58 .
drwxr-xr-x 8 root root 4096 oct 23 17:23 ..
-rw-r--r-- 1 root root 2441 oct 27 18:31 webserver1
```

Y lo configuramos cambiando los siguientes parámetros:

```
sudo nano /etc/nginx/sites-available/webserver1
```

```
albertom-servidor@servidor-debian:~$ sudo nano /etc/nginx/sites-available/webserver1
```

Debemos cambiar la dirección del directorio raíz donde se encuentran los ficheros del sitio web pasando de /var/www/webserver1/

```
root /var/www/webserver1/;
```

También hay que cambiar el server\_name pasando de webserver a webserver1:

```
server_name webserver1;
```

Por último, añadiremos una cabecera personalizada, en nuestro caso será Serv\_Web1\_albertoM:

```
location / {
    add_header Host Serv_Web1_albertoM;
    # First attempt to serve request as file, then
    # as directory, then fall back to displaying a 404.
    try_files $uri $uri/ =404;
}
```

Ahora debemos crear el enlace simbólico en /sites-enabled:

```
sudo ln -s /etc/nginx/sites-available/webserver1 /etc/nginx/sites-enabled/webserver1
```

```
albertom-servidor@servidor-debian:~$ sudo ln -s /etc/nginx/sites-available/webserver1 /etc/nginx/sites-enabled/webserver1
albertom-servidor@servidor-debian:~$ ls -la /etc/nginx/sites-enabled/
total 8
drwxr-xr-x 2 root root 4096 oct 29 07:13 .
drwxr-xr-x 8 root root 4096 oct 23 17:23 ..
lrwxrwxrwx 1 root root   37 oct 29 07:13 webserver1 -> /etc/nginx/sites-available/webserver1
```

Una vez hecho esto reiniciamos el servidor web nginx y comprobamos su estado:

```
sudo systemctl restart nginx.service
```

```
sudo systemctl status nginx.service
```

```
albertom-servidor@servidor-debian:~$ sudo systemctl restart nginx.service
albertom-servidor@servidor-debian:~$ sudo systemctl status nginx.service
● nginx.service - A high performance web server and a reverse proxy server
   Loaded: loaded (/lib/systemd/system/nginx.service; enabled; preset: enabled)
   Active: active (running) since Sun 2023-10-29 07:14:23 CET; 1s ago
     Docs: man:nginx(8)
  Process: 1472 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
  Process: 1473 ExecStart=/usr/sbin/nginx -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
 Main PID: 1474 (nginx)
    Tasks: 3 (limit: 2284)
   Memory: 2.3M
      CPU: 15ms
   CGroup: /system.slice/nginx.service
           └─1474 "nginx: master process /usr/sbin/nginx -g daemon on; master_process on;"
             └─1475 "nginx: worker process"
               └─1476 "nginx: worker process"

oct 29 07:14:23 servidor-debian systemd[1]: Starting nginx.service - A high performance web server and a reverse proxy server...
oct 29 07:14:23 servidor-debian systemd[1]: Started nginx.service - A high performance web server and a reverse proxy server.
```

## CONFIGURACIÓN PREVIA DEL SERVIDOR WEB 2

Para el servidor web 2 debemos hacer una clonación enlazada del primero, pero generando nuevas direcciones MAC para poder tener IP propia.

En este servidor debemos realizar lo siguiente, debemos renombrar el directorio /var/www/webserver1 a /var/www/webserver2:

```
sudo mv /var/www/webserver1 /var/www/webserver2
```

```
albertom-servidor@servidor-debian:~$ sudo mv /var/www/webserver1 /var/www/webserver2
albertom-servidor@servidor-debian:~$ ls -la /var/www
total 12
drwxr-xr-x  3 root    root    4096 oct 28 23:16 .
drwxr-xr-x 12 root    root    4096 oct  6 19:07 ..
drwxr-xr-x  3 www-data www-data 4096 oct 28 21:46 webserver2
```

Vamos a modificar el contenido del index.html para hacer referencia a que este es el servidor web 2:

```
sudo nano /var/www/webserver2/index.html
```

```
albertom-servidor@servidor-debian:~$ sudo nano /var/www/webserver2/index.html
```

```
GNU nano 7.2 /var/www/webserver2/html/index.html
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Prueba de balanceo de carga con Nginx</title>
</head>
<body>
  <h2>Este es el servidor web 2</h2>
  <p>Comprueba el balanceo de carga con Nginx recargando esta página.</p>
</body>
</html>
```

Ahora debemos hacer el unlink del enlace simbólico de /etc/nginx/sites-enabled/ que apunta al fichero webserver1 del directorio /sites-available:

```
albertom-servidor@servidor-debian:~$ ls -la /etc/nginx/sites-enabled/
total 8
drwxr-xr-x  2 root root 4096 oct 29 07:13 .
drwxr-xr-x  8 root root 4096 oct 23 17:23 ..
lrwxrwxrwx  1 root root   37 oct 29 07:13 webserver1 -> /etc/nginx/sites-available/webserver1
```

Para ello hacemos uso del comando unlink:

```
sudo unlink /etc/nginx/sites-enabled/webserver1
```

```
albertom-servidor@servidor-debian:~$ sudo unlink /etc/nginx/sites-enabled/webserver1
albertom-servidor@servidor-debian:~$ ls -la /etc/nginx/sites-enabled/
total 8
drwxr-xr-x  2 root root 4096 oct 29 07:18 .
drwxr-xr-x  8 root root 4096 oct 23 17:23 ..
```

Una vez conseguido esto, vamos a modificar el nombre del archivo de configuración que teníamos pasando de webserver1 a webserver2:

```
sudo mv /etc/nginx/sites-available/webserver1 /etc/nginx/sites-available/webserver2
```

```
albertom-servidor@servidor-debian:~$ sudo mv /etc/nginx/sites-available/webserver1 /etc/nginx/sites-available/webserver2
albertom-servidor@servidor-debian:~$ ls -la /etc/nginx/sites-available/
total 12
drwxr-xr-x  2 root root 4096 oct 29 07:19 .
drwxr-xr-x  8 root root 4096 oct 23 17:23 ..
-rw-r--r--  1 root root 2440 oct 29 07:12 webserver2
```



Y realizamos una serie de cambios en el interior del fichero:

```
sudo nano /etc/nginx/sites-available/webserver2
```

```
albertom-servidor@servidor-debian:~$ sudo nano /etc/nginx/sites-available/webserver2
```

Debemos cambiar los mismos campos que en el servidor 1, es decir, el root, el server\_name y añadir una cabecera personalizada que informe del servidor en el que estamos (en este caso utilizamos Serv\_Web2\_albertoM) para que hagan referencia a los directorios y dirección de este webserver2:

```
root /var/www/webserver2/;

# Add index.php to the list if you are using PHP
index index.html index.htm index.nginx-debian.html;

server_name webserver2;

location / {
    add_header Host Serv_Web2_albertoM;
    # First attempt to serve request as file, then
    # as directory, then fall back to displaying a 404.
    try_files $uri $uri/ =404;
}
```

El puerto de escucha lo debemos mantener en 8080.

Completado esto debemos crear el enlace simbólico en /sites-enabled:

```
sudo ln -s /etc/nginx/sites-available/webserver2 /etc/nginx/sites-enabled
```

```
albertom-servidor@servidor-debian:~$ sudo ln -s /etc/nginx/sites-available/webserver2 /etc/nginx/sites-enabled/
albertom-servidor@servidor-debian:~$ ls -la /etc/nginx/sites-enabled/
total 8
drwxr-xr-x 2 root root 4096 oct 29 07:23 .
drwxr-xr-x 8 root root 4096 oct 23 17:23 ..
lrwxrwxrwx 1 root root   37 oct 29 07:23 webserver2 -> /etc/nginx/sites-available/webserver2
```

Una vez completados estos pasos debemos reiniciar el servidor web y comprobar que su estado es correcto:

```
sudo systemctl restart nginx.service
```

```
sudo systemctl status nginx.service
```

```
albertom-servidor@servidor-debian:~$ sudo systemctl restart nginx.service
albertom-servidor@servidor-debian:~$ sudo systemctl status nginx.service
● nginx.service - A high performance web server and a reverse proxy server
   Loaded: loaded (/lib/systemd/system/nginx.service; enabled; preset: enabled)
   Active: active (running) since Sun 2023-10-29 07:23:54 CET; 5s ago
     Docs: man:nginx(8)
  Process: 1442 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
  Process: 1443 ExecStart=/usr/sbin/nginx -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
 Main PID: 1444 (nginx)
    Tasks: 3 (limit: 2284)
   Memory: 2.3M
      CPU: 15ms
   CGroup: /system.slice/nginx.service
           └─1444 "nginx: master process /usr/sbin/nginx -g daemon on; master_process on;"
              └─1445 "nginx: worker process"
                 └─1446 "nginx: worker process"

oct 29 07:23:54 servidor-debian systemd[1]: Starting nginx.service - A high performance web server and a reverse proxy server...
oct 29 07:23:54 servidor-debian systemd[1]: Started nginx.service - A high performance web server and a reverse proxy server.
```

## CONFIGURACIÓN DEL PROXY INVERSO Y BALANCEADOR DE CARGA

Ya hemos conseguido tener dos servidores web, cada uno con un sitio web personalizado, ahora vamos a configurar nuestro proxy inverso para que funcione como balanceador de carga.

Lo primero que vamos a realizar es eliminar el enlace simbólico a la web estática que usamos en la anterior práctica.

```
albertom-servidor@servidor-debian:~$ ls -la /etc/nginx/sites-enabled/
total 8
drwxr-xr-x 2 root root 4096 oct 28 11:03 .
drwxr-xr-x 8 root root 4096 oct 23 17:23 ..
lrwxrwxrwx 1 root root 35 oct 27 17:07 estatica -> /etc/nginx/sites-available/estatica
```

Para ello haremos volveremos a utilizar el comando unlink:

```
sudo unlink /etc/nginx/sites-enabled/estatica
```

```
albertom-servidor@servidor-debian:~$ sudo unlink /etc/nginx/sites-enabled/estatica
albertom-servidor@servidor-debian:~$ ls -la /etc/nginx/sites-enabled/
total 8
drwxr-xr-x 2 root root 4096 oct 29 07:26 .
drwxr-xr-x 8 root root 4096 oct 23 17:23 ..
```

Una vez hecho esto vamos a renombrar el archivo “estatica” del directorio /sites-available a “balanceo”:

```
sudo mv /etc/nginx/sites-available/estatica /etc/nginx/sites-available/balanceo
```

```
albertom-servidor@servidor-debian:~$ sudo mv /etc/nginx/sites-available/estatica /etc/nginx/sites-available/balanceo
albertom-servidor@servidor-debian:~$ ls -la /etc/nginx/sites-available/
total 12
drwxr-xr-x 2 root root 4096 oct 29 07:27 .
drwxr-xr-x 8 root root 4096 oct 23 17:23 ..
-rw-r--r-- 1 root root 149 oct 27 18:44 balanceo
```

Y cambiamos su interior colocando el siguiente código:

```
GNU nano 7.2 /et
upstream backend_hosts {
    random;
    server webserver1:8080;
    server webserver2:8080;
}

server {
    listen 80;
    server_name www.balanceo.es;
    location / {
        proxy_pass http://backend_hosts;
    }
}
```

El bloque upstream al que hemos llamado backend\_hosts guarda los servidores entre los que se van a repartir las diferentes llamadas que reciba el proxy, en nuestro caso serán los servidores webserver1 y webserver2 los cuales estarán escuchando por sus puertos 8080.

Este bloque lleva un parámetro especial llamado random que es una de las formas posibles que tenemos de realizar un balanceo de carga en Nginx. La forma en la que funciona es muy simple, una vez que llegue una solicitud al proxy este decidirá de forma aleatoria si esa solicitud se destina a webserver1:8080 o a webserver2:8080.

En el bloque server, al igual que hemos hecho en otras prácticas colocamos el puerto de escucha del proxy, el nombre de dominio que tendrá nuestro sitio para que sea fácilmente accesible y en el location / incluimos la directiva proxy\_pass que llamará al bloque upstream para que se aleatorice el servidor que se hará cargo de cada petición que llegue.

Ahora vamos a crear el enlace simbólico a nuestro archivo de configuración en el directorio /sites-enabled:

```
sudo ln -s /etc/nginx/sites-available/balanceo /etc/nginx/sites-enabled/
```

```
albertom-servidor@servidor-debian:~$ sudo ln -s /etc/nginx/sites-available/balanceo /etc/nginx/sites-enabled/
albertom-servidor@servidor-debian:~$ ls -la /etc/nginx/sites-enabled/
total 8
drwxr-xr-x 2 root root 4096 oct 29 08:14 .
drwxr-xr-x 8 root root 4096 oct 23 17:23 ..
lrwxrwxrwx 1 root root   35 oct 29 08:14 balanceo -> /etc/nginx/sites-available/balanceo
```

Una vez completado esto debemos actualizar el fichero /etc/hosts añadiendo las direcciones IP de webserver1 y webserver2 para que el proxy pueda encontrarlos en la red.

```
GNU nano 7.2
127.0.0.1      localhost
127.0.1.1      servidor-debian
10.0.2.11     webserver1
10.0.2.12     webserver2
10.0.2.13     www.balanceo.es
```

Completado todo esto y con ambas máquinas servidoras encendidas ya que de lo contrario recibiríamos un error de host inalcanzable, hay que realizar un reinicio y comprobación de estado del servicio Nginx en nuestra máquina que funciona como proxy inverso/balanceador de carga:

```
sudo systemctl restart nginx.service
```

```
sudo systemctl status nginx.service
```

```
albertom-servidor@servidor-debian:~$ sudo systemctl restart nginx.service
albertom-servidor@servidor-debian:~$ sudo systemctl status nginx
* nginx.service - A high performance web server and a reverse proxy server
   Loaded: loaded (/lib/systemd/system/nginx.service; enabled; preset: enabled)
   Active: active (running) since Sun 2023-10-29 08:18:15 CET; 1s ago
     Docs: man:nginx(8)
  Process: 1498 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
  Process: 1499 ExecStart=/usr/sbin/nginx -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
 Main PID: 1500 (nginx)
   Tasks: 3 (limit: 2284)
    Memory: 2.3M
       CPU: 15ms
    CGroup: /system.slice/nginx.service
            └─1500 "nginx: master process /usr/sbin/nginx -g daemon on; master_process on;"
               └─1501 "nginx: worker process"
                  └─1502 "nginx: worker process"

oct 29 08:18:15 servidor-debian systemd[1]: Starting nginx.service - A high performance web server and a reverse proxy server...
oct 29 08:18:15 servidor-debian systemd[1]: Started nginx.service - A high performance web server and a reverse proxy server.
```

## COMPROBACIÓN DEL FUNCIONAMIENTO DEL BALANCEO DE CARGA

Vamos a realizar una comprobación del funcionamiento, así como de la conexión a los dos servidores haciendo uso de la máquina virtual cliente que tiene Ubuntu instalado.

Lo primero que vamos a necesitar es añadir la IP del proxy a el archivo /etc/hosts de la máquina cliente:

```
GNU nano 6.2
127.0.0.1      localhost
127.0.1.1      albertomcliente-VirtualBox
10.0.2.13      www.balancedo.es
```

### A. AMBOS SERVIDORES OPERATIVOS

Primero vamos a comprobar el funcionamiento del sitio web con ambos servidores operativos.

Si accedemos al sitio web desde el navegador del equipo anfitrión accederemos de forma aleatoria al servidor web 1 o al servidor web 2 (para una doble comprobación podemos acceder a las herramientas de desarrollador y comprobar la cabecera personalizada que recibimos).

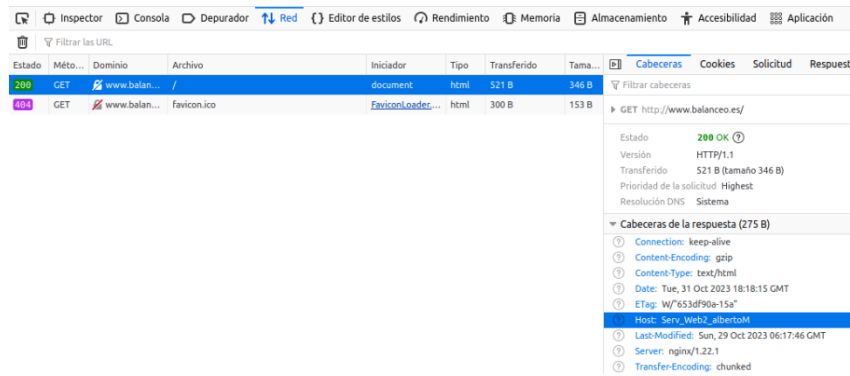
- Servidor web 1:

The screenshot shows a web browser window with the address bar displaying 'www.balancedo.es'. The page content reads 'Este es el servidor web 1' and 'Comprueba el balanceo de carga con Nginx recargando esta página.' Below the browser window, the developer tools are open, showing the 'Cabezas' (Headers) tab. The 'Cabezas de la respuesta' (Response Headers) section is expanded, displaying the following information:

- Estado: 200 OK
- Versión: HTTP/1.1
- Transferido: 522 B (tamaño 346 B)
- Prioridad de la solicitud: Highest
- Resolución DNS: Sistema
- Cabezas de la respuesta (275 B):
  - Connection: keep-alive
  - Content-Encoding: gzip
  - Content-Type: text/html
  - Date: Tue, 31 Oct 2023 18:17:42 GMT
  - ETag: W/"653df6cf-15a"
  - Server: nginx/1.22.1
  - Transfer-Encoding: chunked

- Servidor web 2:

The screenshot shows a web browser window with the address bar displaying 'www.balancedo.es'. The page content reads 'Este es el servidor web 2' and 'Comprueba el balanceo de carga con Nginx recargando esta página.'



Si comprobamos los logs de acceso podremos ver, al igual que en la práctica anterior que el cliente se conecta con el proxy y este se conecta con los servidores 1 y 2.

Para acceder a los logs lo podemos hacer con el siguiente comando:

```
sudo tail /var/log/nginx/access.log
```

- Log de acceso del proxy. En él veremos cómo la IP de conexión es la IP de nuestra máquina Ubuntu:

```
albertom-servidor@servidor-debian:~$ sudo tail /var/log/nginx/access.log
10.0.2.6 - - [31/Oct/2023:19:22:09 +0100] "GET / HTTP/1.1" 304 0 "-" "Moz
10.0.2.6 - - [31/Oct/2023:19:22:09 +0100] "GET / HTTP/1.1" 304 0 "-" "Moz
10.0.2.6 - - [31/Oct/2023:19:22:10 +0100] "GET / HTTP/1.1" 304 0 "-" "Moz
10.0.2.6 - - [31/Oct/2023:19:22:10 +0100] "GET / HTTP/1.1" 304 0 "-" "Moz
10.0.2.6 - - [31/Oct/2023:19:22:10 +0100] "GET / HTTP/1.1" 200 257 "-" "M
10.0.2.6 - - [31/Oct/2023:19:22:11 +0100] "GET / HTTP/1.1" 304 0 "-" "Moz
10.0.2.6 - - [31/Oct/2023:19:22:11 +0100] "GET / HTTP/1.1" 304 0 "-" "Moz
10.0.2.6 - - [31/Oct/2023:19:22:11 +0100] "GET / HTTP/1.1" 304 0 "-" "Moz
10.0.2.6 - - [31/Oct/2023:19:22:11 +0100] "GET / HTTP/1.1" 200 258 "-" "M
10.0.2.6 - - [31/Oct/2023:19:22:12 +0100] "GET / HTTP/1.1" 304 0 "-" "Moz
```

Como podemos ver en la siguiente imagen, la IP de nuestra máquina Ubuntu es efectivamente la 10.0.2.6:

```
albertom-cliente@albertoMartinezPerez:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noq
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu
    link/ether 08:00:27:48:d3:2a brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.6/24 brd 10.0.2.255 scope global
        valid_lft 384sec preferred_lft 384sec
    inet6 fe80::6a7:b4ec:6bbc:f53c/64 scope link
        valid_lft forever preferred_lft forever
```

- Log de acceso del servidor 1. En este log vamos a ver la IP de nuestro proxy (que como vimos en anteriores pasos era la 10.0.2.13):

```
albertom-servidor@servidor-debian:~$ sudo tail /var/log/nginx/access.log
10.0.2.13 - - [31/Oct/2023:19:22:09 +0100] "GET / HTTP/1.0" 200 346 "-" "
10.0.2.13 - - [31/Oct/2023:19:22:09 +0100] "GET / HTTP/1.0" 200 346 "-" "
10.0.2.13 - - [31/Oct/2023:19:22:10 +0100] "GET / HTTP/1.0" 200 346 "-" "
10.0.2.13 - - [31/Oct/2023:19:22:10 +0100] "GET / HTTP/1.0" 304 0 "-" "Mo
10.0.2.13 - - [31/Oct/2023:19:22:10 +0100] "GET / HTTP/1.0" 304 0 "-" "Mo
10.0.2.13 - - [31/Oct/2023:19:22:10 +0100] "GET / HTTP/1.0" 304 0 "-" "Mo
10.0.2.13 - - [31/Oct/2023:19:22:11 +0100] "GET / HTTP/1.0" 304 0 "-" "Mo
10.0.2.13 - - [31/Oct/2023:19:22:11 +0100] "GET / HTTP/1.0" 304 0 "-" "Mo
10.0.2.13 - - [31/Oct/2023:19:22:12 +0100] "GET / HTTP/1.0" 200 346 "-" "
10.0.2.13 - - [31/Oct/2023:19:22:13 +0100] "GET / HTTP/1.0" 304 0 "-" "Mo
```

- Log de acceso del servidor 2. De nuevo en este log veremos como la IP que ha generado solicitudes ha sido la IP de nuestro proxy:

```
albertom-servidor@servidor-debian:~$ sudo tail /var/log/nginx/access.log
10.0.2.13 - - [31/Oct/2023:19:22:05 +0100] "GET / HTTP/1.0" 304 0 "-" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/118.0.0.0 Safari/537.36"
10.0.2.13 - - [31/Oct/2023:19:22:06 +0100] "GET / HTTP/1.0" 200 346 "-" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/118.0.0.0 Safari/537.36"
10.0.2.13 - - [31/Oct/2023:19:22:06 +0100] "GET / HTTP/1.0" 200 346 "-" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/118.0.0.0 Safari/537.36"
10.0.2.13 - - [31/Oct/2023:19:22:07 +0100] "GET / HTTP/1.0" 200 346 "-" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/118.0.0.0 Safari/537.36"
10.0.2.13 - - [31/Oct/2023:19:22:07 +0100] "GET / HTTP/1.0" 304 0 "-" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/118.0.0.0 Safari/537.36"
10.0.2.13 - - [31/Oct/2023:19:22:08 +0100] "GET / HTTP/1.0" 200 346 "-" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/118.0.0.0 Safari/537.36"
10.0.2.13 - - [31/Oct/2023:19:22:09 +0100] "GET / HTTP/1.0" 200 346 "-" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/118.0.0.0 Safari/537.36"
10.0.2.13 - - [31/Oct/2023:19:22:10 +0100] "GET / HTTP/1.0" 304 0 "-" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/118.0.0.0 Safari/537.36"
10.0.2.13 - - [31/Oct/2023:19:22:10 +0100] "GET / HTTP/1.0" 304 0 "-" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/118.0.0.0 Safari/537.36"
10.0.2.13 - - [31/Oct/2023:19:22:11 +0100] "GET / HTTP/1.0" 304 0 "-" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/118.0.0.0 Safari/537.36"
```

## B. SERVIDOR WEB 1 NO OPERATIVO

Vamos a simular una situación donde el servidor web 1 haya sufrido una caída para comprobar que todas las solicitudes a nuestro proxy se derivarán al servidor web 2.

Para ello paralizamos el servidor web 1 con el comando correspondiente de systemctl:

```
sudo systemctl stop nginx.service
```

```
albertom-servidor@servidor-debian:~$ sudo systemctl stop nginx.service
albertom-servidor@servidor-debian:~$ sudo systemctl status nginx.service
nginx.service - A high performance web server and a reverse proxy server
   Loaded: loaded (/lib/systemd/system/nginx.service; enabled; preset: enabled)
   Active: inactive (dead) since Tue 2023-10-31 19:29:50 CET; 4s ago
     Duration: 19min 17.948s
    Docs: man:nginx(8)
   Process: 760 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
   Process: 761 ExecStart=/usr/sbin/nginx -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
   Process: 1415 ExecStop=/sbin/start-stop-daemon --quiet --stop --retry QUIT/5 --pidfile /run/nginx.pid (code=exited, status=0/SUCCESS)
   Main PID: 762 (code=exited, status=0/SUCCESS)
      CPU: 111ms

oct 31 19:10:32 servidor-debian systemd[1]: Starting nginx.service - A high performance web server and a reverse proxy server...
oct 31 19:10:32 servidor-debian systemd[1]: Started nginx.service - A high performance web server and a reverse proxy server.
oct 31 19:29:50 servidor-debian systemd[1]: Stopping nginx.service - A high performance web server and a reverse proxy server...
oct 31 19:29:50 servidor-debian systemd[1]: nginx.service: Deactivated successfully.
oct 31 19:29:50 servidor-debian systemd[1]: Stopped nginx.service - A high performance web server and a reverse proxy server.
```

Ahora si intentamos acceder a la web desde nuestro equipo anfitrión veremos que todas las solicitudes se van a derivar al servidor web 2:



Comprobamos ahora los logs para que veamos que efectivamente ninguna conexión ha podido llegar hasta el servidor 1:

- Log del proxy:

```
albertom-servidor@servidor-debian:~$ sudo tail /var/log/nginx/access.log
10.0.2.6 - - [31/Oct/2023:19:31:31 +0100] "GET / HTTP/1.1" 304 0 "-" "Moz
10.0.2.6 - - [31/Oct/2023:19:31:32 +0100] "GET / HTTP/1.1" 304 0 "-" "Moz
10.0.2.6 - - [31/Oct/2023:19:31:33 +0100] "GET / HTTP/1.1" 304 0 "-" "Moz
10.0.2.6 - - [31/Oct/2023:19:31:34 +0100] "GET / HTTP/1.1" 304 0 "-" "Moz
10.0.2.6 - - [31/Oct/2023:19:31:35 +0100] "GET / HTTP/1.1" 304 0 "-" "Moz
10.0.2.6 - - [31/Oct/2023:19:31:36 +0100] "GET / HTTP/1.1" 304 0 "-" "Moz
10.0.2.6 - - [31/Oct/2023:19:31:37 +0100] "GET / HTTP/1.1" 304 0 "-" "Moz
10.0.2.6 - - [31/Oct/2023:19:31:38 +0100] "GET / HTTP/1.1" 304 0 "-" "Moz
10.0.2.6 - - [31/Oct/2023:19:31:39 +0100] "GET / HTTP/1.1" 304 0 "-" "Moz
10.0.2.6 - - [31/Oct/2023:19:31:40 +0100] "GET / HTTP/1.1" 304 0 "-" "Moz"
```

- Log del servidor 2:

```
albertom-servidor@servidor-debian:~$ sudo tail /var/log/nginx/access.log
10.0.2.13 - - [31/Oct/2023:19:31:30 +0100] "GET / HTTP/1.0" 304 0 "-" "Moz
10.0.2.13 - - [31/Oct/2023:19:31:31 +0100] "GET / HTTP/1.0" 304 0 "-" "Moz
10.0.2.13 - - [31/Oct/2023:19:31:32 +0100] "GET / HTTP/1.0" 304 0 "-" "Moz
10.0.2.13 - - [31/Oct/2023:19:31:33 +0100] "GET / HTTP/1.0" 304 0 "-" "Moz
10.0.2.13 - - [31/Oct/2023:19:31:34 +0100] "GET / HTTP/1.0" 304 0 "-" "Moz
10.0.2.13 - - [31/Oct/2023:19:31:35 +0100] "GET / HTTP/1.0" 304 0 "-" "Moz
10.0.2.13 - - [31/Oct/2023:19:31:36 +0100] "GET / HTTP/1.0" 304 0 "-" "Moz
10.0.2.13 - - [31/Oct/2023:19:31:37 +0100] "GET / HTTP/1.0" 304 0 "-" "Moz
10.0.2.13 - - [31/Oct/2023:19:31:38 +0100] "GET / HTTP/1.0" 304 0 "-" "Moz
10.0.2.13 - - [31/Oct/2023:19:31:39 +0100] "GET / HTTP/1.0" 304 0 "-" "Moz"
```

- Log del servidor 1 (como vemos ninguna de las solicitudes anteriores ha llegado al servidor):

```
albertom-servidor@servidor-debian:~$ sudo tail /var/log/nginx/access.log
10.0.2.13 - - [31/Oct/2023:19:22:09 +0100] "GET / HTTP/1.0" 200 346 "-" "
10.0.2.13 - - [31/Oct/2023:19:22:09 +0100] "GET / HTTP/1.0" 200 346 "-" "
10.0.2.13 - - [31/Oct/2023:19:22:10 +0100] "GET / HTTP/1.0" 200 346 "-" "
10.0.2.13 - - [31/Oct/2023:19:22:10 +0100] "GET / HTTP/1.0" 304 0 "-" "Mo
10.0.2.13 - - [31/Oct/2023:19:22:10 +0100] "GET / HTTP/1.0" 304 0 "-" "Mo
10.0.2.13 - - [31/Oct/2023:19:22:10 +0100] "GET / HTTP/1.0" 304 0 "-" "Mo
10.0.2.13 - - [31/Oct/2023:19:22:11 +0100] "GET / HTTP/1.0" 304 0 "-" "Mo
10.0.2.13 - - [31/Oct/2023:19:22:11 +0100] "GET / HTTP/1.0" 304 0 "-" "Mo
10.0.2.13 - - [31/Oct/2023:19:22:12 +0100] "GET / HTTP/1.0" 200 346 "-" "
10.0.2.13 - - [31/Oct/2023:19:22:13 +0100] "GET / HTTP/1.0" 304 0 "-" "Mo"
```

## C. SERVIDOR WEB 2 NO OPERATIVO

Vamos a hacer lo mismo que en el caso anterior pero ahora sobre el servidor web 2.

Lo primero que haremos será reiniciar el servicio nginx en el servidor web 1:

```
albertom-servidor@servidor-debian:~$ sudo systemctl restart nginx.service
albertom-servidor@servidor-debian:~$ sudo systemctl status nginx.service
● nginx.service - A high performance web server and a reverse proxy server
   Loaded: loaded (/lib/systemd/system/nginx.service; enabled; preset: enabled)
   Active: active (running) since Tue 2023-10-31 19:37:27 CET; 3s ago
     Docs: man:nginx(8)
   Process: 1451 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
   Process: 1452 ExecStart=/usr/sbin/nginx -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
   Main PID: 1453 (nginx)
    Tasks: 3 (limit: 2284)
   Memory: 2.3M
      CPU: 18ms
   CGroup: /system.slice/nginx.service
           └─1453 "nginx: master process /usr/sbin/nginx -g daemon on; master_process on;"
             └─1454 "nginx: worker process"
               └─1455 "nginx: worker process"

oct 31 19:37:27 servidor-debian systemd[1]: Starting nginx.service - A high performance web server and a reverse proxy server...
oct 31 19:37:27 servidor-debian systemd[1]: Started nginx.service - A high performance web server and a reverse proxy server.
```



Y paralizamos el servicio en el servidor 2:

```
albertom-servidor@servidor-debian:~$ sudo systemctl stop nginx.service
albertom-servidor@servidor-debian:~$ sudo systemctl status nginx.service
○ nginx.service - A high performance web server and a reverse proxy server
   Loaded: loaded (/lib/systemd/system/nginx.service; enabled; preset: enabled)
   Active: inactive (dead) since Tue 2023-10-31 19:39:59 CET; 4s ago
     Duration: 29min 34.537s
    Docs: man:nginx(8)
   Process: 754 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
   Process: 755 ExecStart=/usr/sbin/nginx -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
   Process: 1382 ExecStop=/sbin/start-stop-daemon --quiet --stop --retry QUIT/5 --pidfile /run/nginx.pid (code=exited, status=0/SUCCESS)
   Main PID: 756 (code=exited, status=0/SUCCESS)
     CPU: 70ms

oct 31 19:10:22 servidor-debian systemd[1]: Starting nginx.service - A high performance web server and a reverse proxy server...
oct 31 19:10:25 servidor-debian systemd[1]: Started nginx.service - A high performance web server and a reverse proxy server.
oct 31 19:39:59 servidor-debian systemd[1]: Stopping nginx.service - A high performance web server and a reverse proxy server...
oct 31 19:39:59 servidor-debian systemd[1]: nginx.service: Deactivated successfully.
oct 31 19:39:59 servidor-debian systemd[1]: Stopped nginx.service - A high performance web server and a reverse proxy server.
```

Ahora si intentamos acceder al sitio web, todas las solicitudes serán derivadas al servidor web 1:



Y, al igual que antes, vamos a comprobar los logs:

- Log del proxy:

```
albertom-servidor@servidor-debian:~$ sudo tail /var/log/nginx/access.log
10.0.2.6 - - [31/Oct/2023:19:40:36 +0100] "GET / HTTP/1.1" 304 0 "-" "Mo
10.0.2.6 - - [31/Oct/2023:19:40:36 +0100] "GET / HTTP/1.1" 304 0 "-" "Mo
10.0.2.6 - - [31/Oct/2023:19:40:37 +0100] "GET / HTTP/1.1" 304 0 "-" "Mo
10.0.2.6 - - [31/Oct/2023:19:40:38 +0100] "GET / HTTP/1.1" 304 0 "-" "Mo
10.0.2.6 - - [31/Oct/2023:19:40:39 +0100] "GET / HTTP/1.1" 304 0 "-" "Mo
10.0.2.6 - - [31/Oct/2023:19:40:40 +0100] "GET / HTTP/1.1" 304 0 "-" "Mo
10.0.2.6 - - [31/Oct/2023:19:40:41 +0100] "GET / HTTP/1.1" 304 0 "-" "Mo
10.0.2.6 - - [31/Oct/2023:19:40:41 +0100] "GET / HTTP/1.1" 304 0 "-" "Mo
10.0.2.6 - - [31/Oct/2023:19:40:42 +0100] "GET / HTTP/1.1" 304 0 "-" "Mo
10.0.2.6 - - [31/Oct/2023:19:40:43 +0100] "GET / HTTP/1.1" 304 0 "-" "Mo
```

- Log del servidor 1:

```
albertom-servidor@servidor-debian:~$ sudo tail /var/log/nginx/access.log
10.0.2.13 - - [31/Oct/2023:19:40:36 +0100] "GET / HTTP/1.0" 304 0 "-" "Mo
10.0.2.13 - - [31/Oct/2023:19:40:37 +0100] "GET / HTTP/1.0" 304 0 "-" "Mo
10.0.2.13 - - [31/Oct/2023:19:40:37 +0100] "GET / HTTP/1.0" 304 0 "-" "Mo
10.0.2.13 - - [31/Oct/2023:19:40:38 +0100] "GET / HTTP/1.0" 304 0 "-" "Mo
10.0.2.13 - - [31/Oct/2023:19:40:39 +0100] "GET / HTTP/1.0" 304 0 "-" "Mo
10.0.2.13 - - [31/Oct/2023:19:40:40 +0100] "GET / HTTP/1.0" 304 0 "-" "Mo
10.0.2.13 - - [31/Oct/2023:19:40:41 +0100] "GET / HTTP/1.0" 304 0 "-" "Mo
10.0.2.13 - - [31/Oct/2023:19:40:42 +0100] "GET / HTTP/1.0" 304 0 "-" "Mo
10.0.2.13 - - [31/Oct/2023:19:40:43 +0100] "GET / HTTP/1.0" 304 0 "-" "Mo
10.0.2.13 - - [31/Oct/2023:19:40:43 +0100] "GET / HTTP/1.0" 304 0 "-" "Mo
```

- Log del servidor 2 (como vemos ninguna de las solicitudes anteriores ha llegado al servidor):



```
albertom-servidor@servidor-debian:~$ sudo tail /var/log/nginx/access.log
10.0.2.13 - - [31/Oct/2023:19:39:34 +0100] "GET / HTTP/1.0" 200 346 "-"
10.0.2.13 - - [31/Oct/2023:19:39:34 +0100] "GET / HTTP/1.0" 200 346 "-"
10.0.2.13 - - [31/Oct/2023:19:39:34 +0100] "GET / HTTP/1.0" 200 346 "-"
10.0.2.13 - - [31/Oct/2023:19:39:34 +0100] "GET / HTTP/1.0" 200 346 "-"
10.0.2.13 - - [31/Oct/2023:19:39:34 +0100] "GET / HTTP/1.0" 200 346 "-"
10.0.2.13 - - [31/Oct/2023:19:39:34 +0100] "GET / HTTP/1.0" 200 346 "-"
10.0.2.13 - - [31/Oct/2023:19:39:35 +0100] "GET / HTTP/1.0" 200 346 "-"
10.0.2.13 - - [31/Oct/2023:19:39:35 +0100] "GET / HTTP/1.0" 200 346 "-"
10.0.2.13 - - [31/Oct/2023:19:39:35 +0100] "GET / HTTP/1.0" 200 346 "-"
10.0.2.13 - - [31/Oct/2023:19:39:36 +0100] "GET / HTTP/1.0" 200 346 "-"
```

Con estas pruebas hemos comprobado que el funcionamiento de nuestro balanceador de carga es correcto.

## CUESTIONES FINALES

- a) *Busca información de qué otros métodos de balanceo se pueden aplicar con Nginx y describe al menos 3 de ellos.*

En esta práctica hemos utilizado el método random pero tenemos varios más que podemos utilizar:

- 1) **round-robin**: Este método de balanceo de carga es el predeterminado por Nginx y se va a encargar de distribuir las solicitudes entrantes de forma cíclica a través de la lista de servidores que compongan el bloque upstream. Es decir, cada solicitud se enviará al siguiente servidor de la lista en un ciclo continuo.

```
upstream backend_hosts {
    round-robin;
    server webserver1:8080;
    server webserver2:8080;
}
```

NOTA: al ser la opción por defecto no sería necesario colocar la instrucción round-robin en el bloque upstream.

- 2) **least\_conn**: Este otro método va a realizar una distribución de solicitudes de forma que siempre se enviarán al servidor con la menor cantidad de conexiones activas en el momento de la llegada de la solicitud. Es realmente útil cuando los servidores van a tener cargas desiguales.

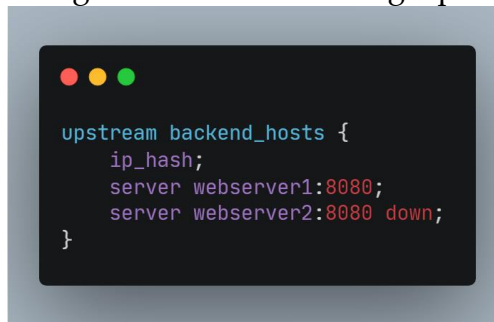
```
upstream backend_hosts {
    least_conn;
    server webserver1:8080;
    server webserver2:8080;
}
```

- 3) *ip\_hash*: En este método se asigna una dirección específica a un servidor, de manera que las solicitudes de la misma IP entrante se destinan al mismo servidor. Este método se utiliza cuando es necesario que haya una persistencia de sesiones.



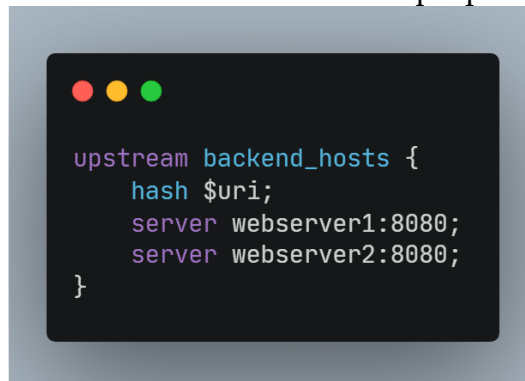
```
upstream backend_hosts {
    ip_hash;
    server webserver1:8080;
    server webserver2:8080;
}
```

Si uno de los servidores debe ser temporalmente parado, se puede marcar con el parámetro "down" con el fin de mantener el hash actual de las direcciones IP de los clientes. En este caso las solicitudes que estaban destinadas a ser procesadas por este servidor se envían automáticamente al siguiente servidor en el grupo.



```
upstream backend_hosts {
    ip_hash;
    server webserver1:8080;
    server webserver2:8080 down;
}
```

- 4) *hash*: En este caso el servidor al que se envía la solicitud se determina a partir de una clave definida por el usuario que puede ser un String, una variable o una combinación. Por ejemplo, la clave puede ser una dirección IP o un URI. Esta forma de balanceo es útil cuando se quiere que unas las solicitudes con ciertas características o rutas se dirijan siempre al mismo servidor. De esta forma cada servidor va a manejar un conjunto de solicitudes basado en el hash proporcionado.



```
upstream backend_hosts {
    hash $uri;
    server webserver1:8080;
    server webserver2:8080;
}
```

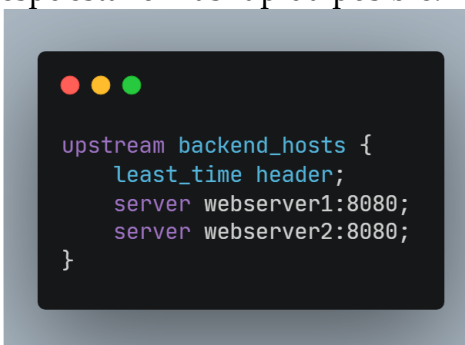
Se puede añadir un parámetro opcional en la directiva hash, el "consistent". Con él se consigue que las solicitudes se distribuyan de

forma equitativa entre todos los servidores del upstream. En el caso de agregar o eliminar un servidor del grupo, sólo se reasignarán unas pocas claves hash consiguiendo con ello minimizar las pérdidas de caché en el caso de servidores de caché con equilibrio de carga o en otras aplicaciones que acumulan estados.



```
upstream backend_hosts {
    hash $uri consistent;
    server webserver1:8080;
    server webserver2:8080;
}
```

- 5) *least\_time*: Por último, vamos a mencionar la opción `least_time` que sólo está disponible en Nginx Plus. Esta se basa en habilitar un balanceador de carga basado en el menor tiempo de respuesta del servidor. Lo que se va a hacer es direccionar las solicitudes hacia los servidores que han tenido un menor tiempo de respuesta en los últimos momentos. Este tipo de balanceo es muy útil cuando se quiere enviar solicitudes a los servidores más rápidos o menos ocupados con el fin de dar una respuesta lo más rápida posible.




```
upstream backend_hosts {
    least_time header;
    server webserver1:8080;
    server webserver2:8080;
}
```

En el header colocaremos la opción que vamos a utilizar para calcular el tiempo que se tarda en recibir el primer byte desde el servidor, por ejemplo, podemos usar la cabecera HTTP Time-To-First-Byte para conseguir esta información.

- b) *Si quiero añadir 2 servidores web más al balanceo de carga, describe detalladamente qué configuración habría que añadir y dónde.*

Tras configurar los directorios web de `/var/www` para que contengan uno de los sitios web previamente creados en esta práctica o dos nuevos que, por ejemplo, destinen al usuario a una nueva web informativa del servidor donde se encuentra deberemos modificar los archivos de configuración de `webserver3` y de `webserver4`:

- webserver3: En el root colocaremos el directorio que hemos creado para almacenar los ficheros del sitio web. Además, añadiremos un nombre de servidor, por ejemplo, webserver3.



```
server {
    listen 8080;
    listen [::]:8080;
    root /var/www/webserver3/;
    server_name webserver3;

    location / {
        add_header Host Serv_Web3_albertoM;
        try_files $uri $uri/ =404;
    }
}
```

- webserver4: Realizaremos los mismos cambios que en el caso del webserver3.



```
server {
    listen 8080;
    listen [::]:8080;
    root /var/www/webserver4/;
    server_name webserver4;

    location / {
        add_header Host Serv_Web4_albertoM;
        try_files $uri $uri/ =404;
    }
}
```

Tras realizar los correspondientes enlaces simbólicos en cada uno de los servidores que apunten a estos archivos de configuración que hemos creado debemos reconfigurar nuestro proxy-balanceador de carga.

Los archivos de configuración del proxy-balanceador quedarían así:

- /etc/nginx/sites-available/balaneo: En este archivo debemos modificar el bloque upstream añadiendo los dos nuevos servidores junto con su puerto de escucha.



```
upstream backend_hosts {
    random;
    server webserver1:8080;
    server webserver2:8080;
    server webserver3:8080;
    server webserver4:8080;
}

server {
    listen 80;
    listen [::]:80;

    server_name www.balaneo.es;

    location / {
        proxy_pass http://backend_hosts;
    }
}
```

De este fichero será necesario crear un enlace simbólico en el directorio /etc/nginx/sites-enabled/.

- /etc/hosts: En este archivo debemos añadir las direcciones IP de los servidores junto con el nombre de dominio que se les ha asignado (por ejemplo, webserver3 y webserver4), de esta forma el proxy cuando tenga que hacer la llamada a, por ejemplo, webserver3 sabrá a qué dirección IP tiene que realizar dicha llamada.



- c) *Describe todos los pasos que deberíamos seguir y configurar para realizar el balanceo de carga con una de las webs de prácticas anteriores.*

*Indicad la configuración de todas las máquinas (webserver, proxy...) y de sus servicios.*

En primer lugar, debemos tener dos servidores que contengan el sitio web que utilizamos para [www.estatica.es](http://www.estatica.es) y que se encuentre en /var/www/ en el caso del primer servidor en un directorio webserver1/ y en el caso del segundo servidor en un directorio webserver2/. Estos directorios deben ser propiedad del usuario gestor del servidor web www-data así que hay que asegurarse de que es propietario de estos directorios (y de todos los ficheros que contienen) y que tiene los permisos necesarios para operar con los ficheros.

Una vez hecho esto debemos configurar cada servidor en el directorio /etc/nginx/sites-available/. En cada uno de ellos crearemos un archivo de configuración (por ejemplo, los podemos llamar webserver1 y webserver2 respectivamente) que contenga la siguiente configuración.

- Servidor 1:



- Servidor 2:



```
server {
    listen 8080;
    listen [::]:8080;

    root /var/www/webserver2;

    server_name webserver2;

    location / {
        try_files $uri $uri/ =404;
    }
}
```

Una vez completada esta configuración debemos crear los enlaces simbólicos a estos ficheros en el directorio `/etc/nginx/sites-available/`.

Completado esto debemos realizar una configuración de nuestro proxy inverso que funcionará como balanceador de carga.

En este servidor debemos crear el siguiente archivo en `/etc/nginx/sites-available/`



```
upstream backend_hosts {
    random;
    server webserver1:8080;
    server webserver2:8080;
}

server {
    listen 80;
    listen [::]:80;

    server_name www.estatica.es;

    location / {
        proxy_pass http://backend_hosts;
    }
}
```

Una vez hecho el archivo debemos crear el enlace simbólico correspondiente en `/etc/nginx/sites-enabled/`.

Junto con esto deberemos modificar el archivo `/etc/hosts` para que el proxy sea capaz de encontrar a los dos servidores que se van a repartir las peticiones:



```
127.0.0.1          localhost
127.0.1.1          servidor-debian
direccion_ip_proxy  www.estatica.es
direccion_ip_webserver1  webserver1
direccion_ip_webserver2  webserver2
```

Con esto tendríamos nuestro sistema configurado para poder servir el sitio web `www.estatica.es` a través de un proxy inverso que actuara de balanceador de carga entre dos servidores que alojarían el sitio.