

# Informe Proyecto Final “Balanceo de Carga de Servidores Web”

Alejandra Garcés Gil  
Universidad Autónoma de  
Occidente  
Cali, Colombia  
[Alejandra.garces@uao.edu.co](mailto:Alejandra.garces@uao.edu.co)

Juan Esteban Lizarazo  
Quintero  
Universidad Autónoma de  
Occidente  
Cali, Colombia  
[juan.lizarazo@uao.edu.co](mailto:juan.lizarazo@uao.edu.co)

Victor Andrés Silva Moreno  
Universidad Autónoma de  
Occidente  
Cali, Colombia  
[victor.silva@uao.edu.co](mailto:victor.silva@uao.edu.co)

## II. Marco Teórico

### 1. Docker

Docker es una plataforma de código abierto que facilita la creación, implementación y ejecución de aplicaciones en contenedores. Los contenedores son entornos ligeros y portátiles que contienen todo lo necesario para ejecutar una aplicación, incluidas las bibliotecas, dependencias y código. Docker utiliza la virtualización a nivel de sistema operativo para aislar las aplicaciones y garantizar que funcionen de manera consistente en cualquier entorno.

### 2. Apache Jmeter

Apache JMeter es una herramienta de prueba de carga y rendimiento de código abierto desarrollada por la Apache Software Foundation. Se utiliza para medir el rendimiento y la carga de aplicaciones web, bases de datos, servidores FTP y otros servicios. JMeter permite simular un gran número de usuarios concurrentes para evaluar cómo una aplicación o servidor se comporta bajo diferentes condiciones de carga.

### 3. HAProxy

En sí, HAProxy significa High Availability Proxy, es un software de código abierto que proporciona balanceo de carga y capacidades de proxy para aplicaciones web. Su función principal es distribuir el tráfico de red entrante entre varios servidores backend para mejorar la disponibilidad y el rendimiento de las aplicaciones. HAProxy es ampliamente

**Resumen:** En este documento se relata el desarrollo del proyecto final correspondiente al balanceo de cargas por medio de Proxy, buscando una distribución balanceada de peticiones dentro de, en este caso específico, dos máquinas virtuales receptoras. Se tomaron en cuenta dos alternativas de distribución, cada una con su nivel de funcionamiento, rendimiento y comportamiento, ambas siendo probadas de forma constante en diferentes niveles de “esfuerzo”.

**~Abstract:** This document describes the development of the final project corresponding to load balancing through proxy, seeking a balanced distribution of requests within, in this specific case, two receiving virtual machines. Two distribution alternatives were considered, each with its level of operation, performance, and behaviour, both being tested consistently at different levels of "effort".

**Keywords:** Docker, Docker.engine, Docker IO, JMeter, HAProxy, Workers, Artillery, mod\_proxy\_balancer, Frontend, backend.

## I. Introducción

El propósito de esta práctica es implementar un clúster de servidores web por medio de un balanceo de cargas. Idealmente, el funcionamiento estará centrado en el evitar una saturación de las máquinas dentro del clúster creado, distribuyendo correctamente las peticiones generadas en cada inyección, entre las integrantes de éste, siendo cada servidor capaz de resolverlas. Se debe dejar claro igual que la petición no estará ligada a los servidores, y será el propio balanceador el que decidirá a cuál de las máquinas redirigir cada una de las peticiones.

utilizado en entornos de producción para mejorar la confiabilidad y escalabilidad de las aplicaciones web. Puede integrarse con diversas tecnologías y es especialmente popular como un componente central en arquitecturas de aplicaciones distribuidas

#### 4. Nginx

Nginx (pronunciado "engine-x") es un servidor web de código abierto y un servidor proxy inverso. Además de sus funciones como servidor web y proxy, Nginx también puede actuar como equilibrador de carga, servidor de correo electrónico (para protocolos como IMAP, POP3, y SMTP), y como servidor para aplicaciones basadas en WebSocket. Fue creado por Igor Sysoev y lanzado por primera vez en 2004. Nginx es conocido por su rendimiento, escalabilidad y eficiencia en el manejo de grandes cantidades de tráfico web. Es ampliamente utilizado como servidor web en sitios de alto tráfico y en arquitecturas de aplicaciones modernas. Su popularidad se debe en parte a su diseño ligero y a su capacidad para manejar eficientemente una variedad de tareas relacionadas con el servidor web y el proxy inverso.

### III. Implementación

#### 1. Alternativa Número 1: Balanceo de cargas con configuración nginx

El balanceo de cargas con configuración Nginx es una técnica que permite distribuir la carga de trabajo entre varios servidores web. Esto se hace con el fin de mejorar el rendimiento, la disponibilidad y la capacidad de escalabilidad de una aplicación o sitio web. Nginx ofrece un módulo llamado upstream que permite configurar el balanceo de cargas. El módulo upstream se utiliza para definir un grupo de

servidores web que se utilizarán para manejar las solicitudes.

Para configurar el balanceo de cargas con Nginx, se deben realizar los siguientes pasos; el primer paso es crear un grupo de servidores web, el segundo definir un algoritmo de balanceo de cargas y por último asignar el grupo de servidores web a una ubicación.

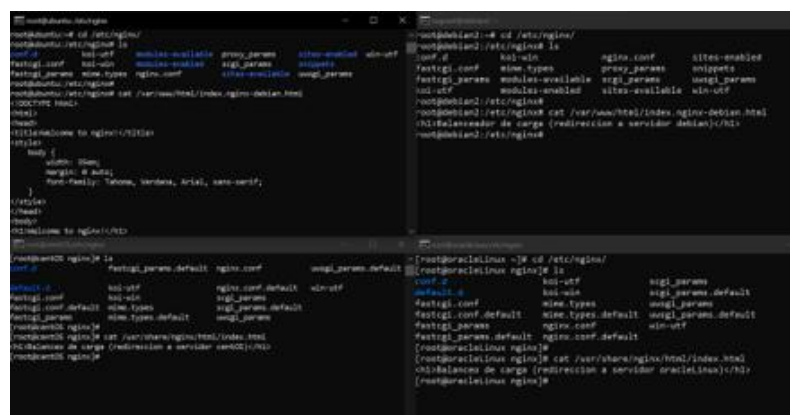
A continuación, se explicará brevemente paso a paso la configuración necesaria para implantar el balanceo de cargas; lo primero es instalar las librerías de Nginx y HTTP con los siguientes comandos.

```
sudo dnf install nginx
```

```
sudo dnf install httpd
```

```
sudo systemctl start httpd
```

Este paso se debe realizar en todos los servidores que se vayan a configurar; el siguiente paso es entrar a la dirección /etc/nginx, y crear un archivo html para configurar la página; en la siguiente imagen se encuentra como debe quedar configurado.

The image shows two terminal windows side-by-side. The left window shows the installation of Nginx and the creation of a default configuration file. The right window shows the configuration of the Nginx upstream module to balance load between two servers. The commands shown are: 

```
sudo dnf install nginx
sudo dnf install httpd
sudo systemctl start httpd
cd /etc/nginx
cat /usr/share/nginx/html/index.html
cat /etc/nginx/conf.d/default.conf
cat /etc/nginx/conf.d/default.conf
```

Después de tener configuradas las paginas y las maquinas el paso siguiente es hacer el balanceos de las cargas, para esto se debe escoger una de las máquinas y realizar el siguiente paso, primero se elimina la configuración que viene por defecto en el sites-enable del serivodr ngx, esto se hace usando el siguiente comando.

Vim /sites-enabled rm default

```
root@ubuntu: /etc/nginx/sites-enabled
root@ubuntu:~# cd /etc/nginx/
root@ubuntu:/etc/nginx# ls
conf.d      fastcgi_params  koi-win        modules-enabled  nginx.conf  scgi
fastcgi.conf  koi-utf        mime.types     modules-enabled  proxy_params  ssl
root@ubuntu:/etc/nginx# cd sites-enabled/
root@ubuntu:/etc/nginx/sites-enabled# rm default
```

El siguiente paso es crear un archivo en la ruta /etc/nginx/conf.d para configurar el balanceo de cargas del servidor; el balanceador de carga recibe las solicitudes de los clientes y las redirige a uno de los servidores web del grupo. Para ello, utiliza el algoritmo de planificación de Round Robin, que consiste en ir rotando las solicitudes entre los servidores web del grupo de forma uniforme, en concreto, el balanceador debe conocer las direcciones IP y los puertos de los servidores web del grupo. Además, debe especificar la ruta por la que se accederá a los servidores web.

Continuación encontramos el algoritmo que permite hacer el direccionamiento de las IPs del servidor.

```
upstream backend {
server 192.168.1.100:80;
server 192.168.1.101:80;}
server { listen 80;
        server_name example.com;
        location /
{    proxy_pass http://backend;    }}
```

Con esta configuración, la primera solicitud que reciba el balanceador se redirigirá al servidor web con la dirección IP 192.168.1.100. La siguiente solicitud se redirigirá al servidor web con la dirección IP 192.168.1.101, y así sucesivamente; con esta configuración ya queda listo para el balanceo de cargas, solo faltaría reiniciar el servicio con el siguiente comando

systemctl reload nginx' y probar directamente desde el navegador.

## 2. Alternativa Número 2: HAProxy como servidor Web

La configuración de Haproxy se realiza a través de un archivo de configuración, donde se especifica el puerto en el que escuchara el equilibrio de carga, dirección IP del frontend, lista de servidores que van a procesar las solicitudes. Se explora la implementación de balanceadores de carga en un entorno basado en CentOS 9s, con instrucciones específicas para configurar el módulo Proxy Balancer de Apache. En sí, podemos determinar al Proxy Balancer como un algoritmo de balanceo de carga configurable, y permitirá la distribución de solicitudes de clientes entre múltiples servidores backend, optimizando el rendimiento y la disponibilidad de un sitio web o aplicación.

### - Implementación

Configuración del VagrantFile

### - Instalación del módulo "mod\_proxy\_balancer"

Se utiliza el siguiente comando para realizar la instalación de este módulo

"sudo dnf install httpd httpd-tools mod\_ssl"

### - Configuración del módulo para habilitar el proxy\_balancer

Al momento de querer configurar este módulo, debemos editar el siguiente documento

"sudo vim /etc/httpd/conf/httpd.conf"

Ya en ese archivo, debemos añadir las siguientes líneas, utilizados principalmente para cargar módulos específicos relacionados al proxy y el equilibrado de cargas

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

Vagrant.configure("2") do |config|
  if Vagrant.has_plugin? "vagrant-vbguest"
    config.vbguest.no_install = true
    config.vbguest.auto_update = false
    config.vbguest.no_remote = true
  end

  config.vm.define :servidorRest do |servidorRest|
    servidorRest.vm.box = "generic/centos9s"
    servidorRest.vm.network :private_network, ip: "192.168.60.3"
    servidorRest.vm.hostname = "servidorRest"
  end

  config.vm.define :servidor1 do |servidor1|
    servidor1.vm.box = "generic/centos9s"
    servidor1.vm.network :public_network
    servidor1.vm.network :private_network, ip: "192.168.60.4"
    servidor1.vm.hostname = "servidor1"
  end

  config.vm.define :servidor2 do |servidor2|
    servidor2.vm.box = "generic/centos9s"
    servidor2.vm.network :private_network, ip: "192.168.60.2"
    servidor2.vm.hostname = "servidor2"
  end
end
```

LoadModule proxy\_balancer\_module  
modules/mod\_proxy\_balancer.so  
(encargado del equilibrado de carga)

LoadModule proxy\_module  
modules/mod\_proxy.so (proporciona  
funcionalidad básica de un proxy inverso, y  
es utilizado para redirigir solicitudes a  
servidores backend, mejorando el  
rendimiento y su disponibilidad)

LoadModule proxy\_http\_module  
modules/mod\_proxy\_http.so  
(utilizado para admitir el protocolo HTTP,  
y permitirá que Apache actúe como un  
proxy HTTP, manejando solicitudes y  
respuestas bajo el mismo protocolo entre  
clientes y servidores backend)

```
Include conf.modules.d/*.conf
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_http_module modules/mod_proxy_http.so
LoadModule proxy_balancer_module modules/mod_proxy_balancer.so
```

```
<Proxy balancer://mycluster>
  BalancerMember http://10.0.0.1:80
  BalancerMember http://10.0.0.2:80
</Proxy>

ProxyPass /test balancer://mycluster
ProxyPassReverse /test balancer://mycluster
```

```
<Proxy balancer://mycluster>
  BalancerMember http://192.168.60.4:80
  BalancerMember http://192.168.60.2:80
  ProxySet lbmethod=bytraffic
</Proxy>

<VirtualHost *:80>
  ServerName myserver.com
  ProxyPreserveHost On
  ProxyPass / balancer://mycluster/
  ProxyPassReverse / balancer://mycluster/

  DocumentRoot /var/www/html
  <Directory /var/www/html>
    AllowOverride All
    Require all granted
    DirectoryIndex main.html index.html
  </Directory>
</VirtualHost>
```

Acá hay una vista “final” de cómo queda la configuración



Habiendo terminado la configuración,  
procedemos a la implementación y prueba  
completa:

#### Balanceo de Cargas

- [Qué es?](#)
- [Cómo funciona?](#)
- [Beneficios](#)

#### ¿Qué es el balanceo de cargas?

El balanceo de cargas es una técnica utilizada en la infraestructura de servidores para distribuir el tráfico de red de manera uniforme entre varios servidores. Su objetivo es evitar la sobrecarga de un servidor en particular y mejorar la disponibilidad y el rendimiento del sistema.

#### ¿Cómo funciona el balanceo de cargas?

El balanceo de cargas distribuye las solicitudes de los usuarios entre varios servidores en un grupo. Un dispositivo o software de balanceo de cargas evalúa el estado de los servidores y dirige las solicitudes al servidor más adecuado en función de varios criterios, como la carga actual, la disponibilidad y otros factores.

#### Beneficios del balanceo de cargas

- Mejora la disponibilidad y la confiabilidad del servicio.
- Evita la carga de servidores individuales.
- Mejora el rendimiento y la velocidad de respuesta.
- Facilita la escalabilidad horizontal al agregar nuevos servidores según sea necesario.

© 2023 Tu Sitio Web de Balanceo de Cargas

Al momento de recargar la página, el  
propio distribuidor hará el cambio y otro de  
los servidores recibirá la petición

## Balaneo de cargas 2

- [Objetivo](#)
- [Como funciona](#)
- [Beneficio](#)

### ¿Qué es el balanceo de cargas?

El balanceo de cargas es una técnica utilizada en la infraestructura de servidores para distribuir el tráfico de red de manera uniforme entre varios servidores. Su objetivo es evitar la sobrecarga de un servidor en particular y mejorar la disponibilidad y el rendimiento del sistema.

### como funciona balanceo de cargas?

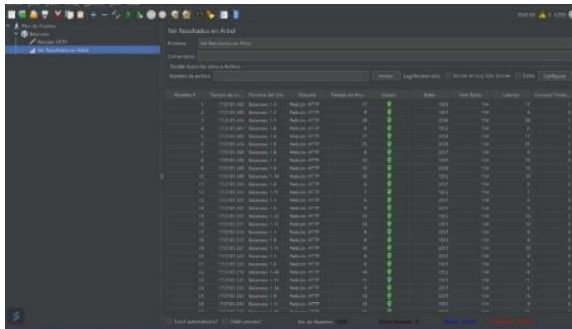
El balanceo de cargas es la distribución de las solicitudes de los usuarios entre varios servidores en un grupo. Un dispositivo o software de balanceo de cargas evalúa el estado de los servidores y dirige las solicitudes entrantes al servidor más adecuado en función de varios criterios, como la carga actual, la disponibilidad y otros factores.

### Beneficio del balanceo de cargas

- Mejora la disponibilidad y la confiabilidad del servicio.
- Evita la carga de servidores individuales.
- Mejora el rendimiento y la velocidad de respuesta.
- Facilita la escalabilidad horizontal al agregar nuevos servidores según sea necesario.

© 2023 Es Sitio Web de Balanceo de Cargas

Ya con este confirmamos, por medio de JMeter, la correcta respuesta del balanceador



Sample	Time	Latency	Response	Size	Code	Message	Success	Errors	Warnings
1	11.000	100	200	1024	200	OK	True	0	0
2	11.001	100	200	1024	200	OK	True	0	0
3	11.002	100	200	1024	200	OK	True	0	0
4	11.003	100	200	1024	200	OK	True	0	0
5	11.004	100	200	1024	200	OK	True	0	0
6	11.005	100	200	1024	200	OK	True	0	0
7	11.006	100	200	1024	200	OK	True	0	0
8	11.007	100	200	1024	200	OK	True	0	0
9	11.008	100	200	1024	200	OK	True	0	0
10	11.009	100	200	1024	200	OK	True	0	0
11	11.010	100	200	1024	200	OK	True	0	0
12	11.011	100	200	1024	200	OK	True	0	0
13	11.012	100	200	1024	200	OK	True	0	0
14	11.013	100	200	1024	200	OK	True	0	0
15	11.014	100	200	1024	200	OK	True	0	0
16	11.015	100	200	1024	200	OK	True	0	0
17	11.016	100	200	1024	200	OK	True	0	0
18	11.017	100	200	1024	200	OK	True	0	0
19	11.018	100	200	1024	200	OK	True	0	0
20	11.019	100	200	1024	200	OK	True	0	0
21	11.020	100	200	1024	200	OK	True	0	0
22	11.021	100	200	1024	200	OK	True	0	0
23	11.022	100	200	1024	200	OK	True	0	0
24	11.023	100	200	1024	200	OK	True	0	0
25	11.024	100	200	1024	200	OK	True	0	0
26	11.025	100	200	1024	200	OK	True	0	0
27	11.026	100	200	1024	200	OK	True	0	0
28	11.027	100	200	1024	200	OK	True	0	0
29	11.028	100	200	1024	200	OK	True	0	0
30	11.029	100	200	1024	200	OK	True	0	0

HAProxy, conjunto a Jmeter, siendo este último una herramienta de prueba de carga y rendimiento cuya principal función es evaluar el rendimiento de aplicaciones web y servidores, nos dan una idea muy clara del funcionamiento, nivel de rendimiento y comportamiento del balanceador, donde el propio JMeter jugó un papel fundamental al momento de querer decidir cuál será la implementación o idea final.

## 3. Alternativa 3: Docker

Para la configuración de Docker iniciamos creando 3 máquinas virtuales, editando el archivo de vagrantfile de la siguiente manera:

Iniciamos máquinas e iniciamos la instalación de Docker con la instalación de plugins y librerías necesarias en todas las máquinas

```
sudo apt update
sudo apt install docker.io
```

```
Vagrant.configure("2") do |config|

  if Vagrant.has_plugin?("vagrant-vbguest")
    config.vbguest.no_install = true
    config.vbguest.auto_update = false
    config.vbguest.no_remote = true
  end

  config.vm.define :cliente1 do |cliente1|
    cliente1.vm.box = "bento/ubuntu-22.04"
    cliente1.vm.network :private_network, ip: "192.168.100.3"
    cliente1.vm.hostname = "cliente1"
    cliente1.vm.box_download_insecure = true
  end

  config.vm.define :cliente2 do |cliente2|
    cliente2.vm.box = "bento/ubuntu-22.04"
    cliente2.vm.network :private_network, ip: "192.168.100.4"
    cliente2.vm.hostname = "cliente2"
    cliente2.vm.box_download_insecure = true
  end

  config.vm.define :servidor1 do |servidor1|
    servidor1.vm.box = "bento/ubuntu-22.04"
    servidor1.vm.network :private_network, ip: "192.168.100.2"
    servidor1.vm.hostname = "servidor1"
    servidor1.vm.box_download_insecure = true
  end

end
```

```
sudo systemctl start docker
sudo systemctl enable Docker
```

Instalamos Docker compose

```
sudo apt install docker-compose
```

Instalación de haproxy

```
sudo apt install haproxy
```

Instalación de apache

```
sudo apt install docker-compose
sudo apt install haproxy
sudo apt install apache2
```

Creamos una carpeta llamada Appbalanceada

```
Mkdir AppBalanceada
```

Entramos en la carpeta y clonamos el repositorio del github para iniciar la configuración

```
git clone
https://github.com/omondragon/haproxy-docker
```



Editamos los archivos index.html disponible en cada uno de los contenedores web

```
Vim /AppBalanceada/haproxy-docker/web/html1/index.html
```

```
Vim /AppBalanceada/haproxy-docker/web/html2/index.html
```

Iniciamos el servicio de Docker ejecutando el siguiente comando

```
Docker-compose up
```

Para la verificación del servicio abrimos una ventana en el navegador y agregamos

<http://192.168.100.2:5080>

Para verificar que el balanceo esta configurado correctamente y se están balanceando las peticiones de manera adecuada a cada uno de los servidores.

<http://192.168.100.2:5080/haproxy?stats>

login: admin  
password: admin

#### 4. Tabla Comparativa

Característica	Proxy balancer	Nginx	Docker
Tipo de software	Servidor web	Servidor web	Contenedores
Instalación	Requiere instalación del módulo proxy balancer en el servidor web	Requiere instalación del servidor web	No requiere instalación de software adicional
Configuración	Se realiza mediante directivas del módulo proxy balancer	Se realiza mediante un archivo de configuración	Se realiza mediante un archivo de configuración
Escalabilidad	Permite escalar el número de servidores web de forma independiente del número de servidores proxy balancer	Permite escalar el número de servidores web y servidores proxy balancer de forma independiente	Permite escalar el número de servidores web y servidores proxy balancer de forma independiente
Disponibilidad	Requiere que el servidor proxy balancer esté disponible	Requiere que los servidores web y el servidor proxy balancer estén disponibles	Requiere que los servidores web y el servidor proxy balancer estén disponibles
Seguridad	Permite aplicar políticas de seguridad al balanceo de carga	Permite aplicar políticas de seguridad al balanceo de carga	Permite aplicar políticas de seguridad al balanceo de carga
Coste	Requiere licencia del software del servidor web	Requiere licencia del software del servidor web	No requiere licencia de software

#### 5. Conclusiones

- El proxy balancer es una solución sencilla y escalable para el balanceo de carga de servidores web. Es una buena opción para entornos donde se requiere una solución de balanceo de carga rápida y sencilla de implementar.
- Nginx es un servidor web potente y flexible. Es una buena opción para entornos donde se requiere un servidor web con funciones avanzadas, como el balanceo de carga.
- Docker es una solución más flexible y adaptable para el balanceo de carga de servidores web. Es una buena opción para entornos donde se requiere una solución de balanceo de carga escalable y segura.

Proxy balancer

Ventajas:

- Es una solución sencilla y fácil de implementar.
- Permite aplicar políticas de seguridad al balanceo de carga.

Desventajas:

- Requiere instalación del módulo proxy balancer en el servidor web.
- La configuración puede ser compleja.

Nginx

Ventajas:

- Es un servidor web potente y flexible.
- Permite aplicar políticas de seguridad al balanceo de carga.

Desventajas:

- Puede ser más complejo de configurar que el proxy balancer

Docker

Ventajas:

- Es una solución más flexible y adaptable.
- Permite escalar el número de servidores web y servidores proxy balancer de forma independiente.
- No requiere licencia de software.

Desventajas:

- Puede ser más complejo de configurar que el proxy balancer o Nginx.
- Requiere un mayor nivel de conocimientos en contenedores.

## 6. Referencias

<https://www.computerworld.es/tendencias/que-es-el-balanceo-de-carga>

[https://www.ibm.com/docs/en/i/7.1?topic=ssw\\_ibm\\_i\\_71/rzaie/rzaiemod\\_proxy\\_balancer.html](https://www.ibm.com/docs/en/i/7.1?topic=ssw_ibm_i_71/rzaie/rzaiemod_proxy_balancer.html)

[https://httpd.apache.org/docs/2.4/mod/mod\\_proxy.html](https://httpd.apache.org/docs/2.4/mod/mod_proxy.html).

[https://httpd.apache.org/docs/2.4/mod/mod\\_proxy\\_balancer.html](https://httpd.apache.org/docs/2.4/mod/mod_proxy_balancer.html).

[https://httpd.apache.org/docs/2.4/mod/mod\\_lbmethod\\_bytraffic.html](https://httpd.apache.org/docs/2.4/mod/mod_lbmethod_bytraffic.html).