

# **BALANCEADOR DE CARGA USANDO EL MÓDULO DE APACHE MOD\_PROXY\_BALANCER + PRUEBAS DE CARGA CON JMeter**

**David Forero Gonzalez - 2151052**  
**Esteban Pardo Jimenez - 2176059**  
**Fabian Andres Beltran - 2156734**  
**Javier Andres Lopez - 2166930**

**Universidad Autónoma de Occidente**

## ***Abstract***

*The Apache HTTP Server, a well-known and widely used web server, is equipped with the mod\_proxy\_balancer module for load balancing across multiple backend servers. To ensure proper and efficient functioning, it is essential to conduct comprehensive testing using a combination of mod\_proxy\_balancer testing and JMeter load testing.*

*In mod\_proxy\_balancer testing, it is crucial to test different scenarios, including failover and load balancing algorithms, to ensure that the module can handle varying types of traffic and server configurations. Additionally, testing the module's ability to handle large volumes of traffic and scale up or down as needed is important.*

*JMeter load testing is used to simulate real-world traffic and assess the module's performance under different scenarios. This helps identify potential bottlenecks, performance issues, and other problems that may arise during peak traffic periods.*

*Apache HTTP Server has several functions, and additional functions can be added using Apache modules. In this project, a load balancing cluster of two or more web servers is used to balance the load. The mod\_proxy module is used to configure Apache as a load balancer, and JMeter is used to test the proxy and verify the proper functioning of Apache.*

## ***I. Introducción***

En el siguiente documento presentaremos los procesos, prácticas y pruebas que realizamos para llevar a cabo el proyecto anteriormente seleccionado abarcando desde instalaciones a configuraciones y en lo posible brindar la información lo más puntual posible, mostraremos cómo funciona un Balanceador

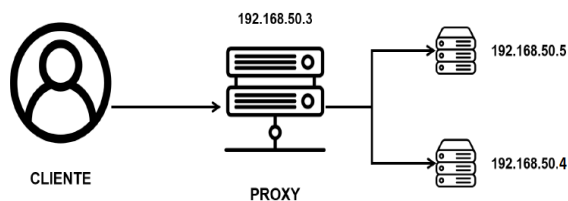
de carga usando el módulo de apache mod\_proxy\_balancer con pruebas de carga con JMeter también una solución a la problemática de balanceo de cargas y unas cuantas breves soluciones distintas a la que implementaremos.

La estructura que usaremos en el proyecto es un servidor proxy inverso el cual es un tipo de intermediario que recibe solicitudes HTTP y las redirige de manera transparente hacia uno o más servidores backend. Su función principal es evitar el acceso directo a los servidores de aplicaciones subyacentes. Además, también se utilizan para distribuir la carga de las solicitudes entrantes entre varios servidores de aplicaciones, lo que mejora el rendimiento y brinda protección contra posibles fallos.

Esta guía proporciona instrucciones sobre cómo configurar Apache como un proxy inverso básico utilizando la extensión mod\_proxy. Esto permitirá redirigir las conexiones entrantes hacia múltiples servidores backend que se encuentren en la misma red.

## ***II. Descripción del problema***

Como primer paso definimos la topología básica de la red, donde vemos las máquinas backend, el proxy y en qué orden interactúan las unas con las otras pondremos en cada una su respectiva dirección ip. Ver figura #1.



**Figura # 1**

Nuestra estructura de red está conformada por dos máquinas virtuales que cuentan con el servicio apache HTTPD como servidores web, los cuales alojan una página web. Estas máquinas se denominan servidor1 y servidor2. Además, existe otra máquina virtual importante llamada proxy, que cumple la función de balanceador de carga. Es decir, este servidor proxy se encarga de distribuir equitativamente las solicitudes HTTP entre los dos servidores web, mejorando así la disponibilidad y el correcto funcionamiento de la aplicación web alojada en los servidores mencionados previamente.

### III. Herramientas

- **VirtualBox:** Es una aplicación que te permite crear y ejecutar máquinas virtuales con diferentes sistemas operativos dentro de tu propio equipo. Esto significa que puedes tener un ordenador con un sistema operativo específico y, al mismo tiempo, crear y utilizar una o varias máquinas virtuales con otros sistemas operativos. Puedes descargar VirtualBox desde este enlace: <https://www.virtualbox.org/>

- **Vagrant:** Es una herramienta que te permite crear y gestionar entornos de máquinas virtuales de manera sencilla y eficiente. Está diseñada para automatizar y agilizar el proceso de configuración de entornos de desarrollo. Puedes descargar Vagrant desde este enlace: <https://www.vagrantup.com/>

- **JMeter:** Es una aplicación de código abierto que se utiliza para realizar pruebas de carga y medir el rendimiento de sistemas. En el entorno implementado, se utilizará JMeter para llevar a cabo pruebas funcionales y evaluar el rendimiento del sistema.

Configuración de las máquinas:  
Vagrant file con aprovisionamiento.  
Ver figura# 2

```

Vagrantfile (2): Bloc de notas
Archivo Edición Formato Ver Ayuda
Vagrant.configure("2") do |config|
  config.vm.boot_timeout = 600
  if Vagrant.has_plugin? "vagrant-vbguest"
    config.ssh.username = 'vagrant'
    config.ssh.password = 'vagrant'
    config.ssh.insert_key = 'true'
    config.vbguest.no_install = true
    config.vbguest.auto_update = false
    config.vbguest.no_remote = true
  end

  config.vm.define :servidor1 do |servidor1|
    config.vm.box = "bento/centos-stream-9"
    servidor1.vm.network :private_network, ip: "192.168.50.3"
    servidor1.vm.hostname = "servidor1"
    servidor1.vm.provision "shell", path: "./config/server_conf.sh"
  end

  config.vm.define :servidor2 do |servidor2|
    config.vm.box = "bento/centos-stream-9"
    servidor2.vm.network :private_network, ip: "192.168.50.4"
    servidor2.vm.hostname = "servidor2"
    servidor2.vm.provision "shell", path: "./config/server_conf.sh"
  end

  config.vm.define :proxy do |proxy|
    config.vm.box = "bento/centos-stream-9"
    proxy.vm.network :private_network, ip: "192.168.50.5"
    proxy.vm.hostname = "proxy"
    proxy.vm.provision "shell", path: "./config/proxy_conf.sh"
  end
end

```

**Figura #2**

Luego de configurar nuestras máquinas virtuales e iniciarlas, nos logueamos con el superusuario para instalar los servicios que son requeridos en nuestro proyecto, inicialmente fueron el apache httpd y la herramienta vim para editar nuestros archivos de configuración, tenemos en cuenta que 2 de las máquinas virtuales creadas, alojan dos sitios webs, siendo estas dos máquinas el servidor1 y el servidor2.

**yum install httpd -y**  
**yum install vim -y**

Configurar index.html de ambas máquinas servidor. Ver Figura #3

```
root@servidor1:/var/www/html
1<DOCTYPE html>
<html>
<head>
<title>Pirate Page</title>
</head>
<style>
body {
background-color: lightblue;
font-family: Arial, background-color:lightblue;
}
h1 {
color:white;
margin-top: 50px;}
</style>
<body>
<header>
<h1>Welcome to the Pirate Page</h1>
</header>
<main>
<section id="about">
<h2>About the Pirate Page</h2>
<p>Welcome to the Pirate Page, where you can learn all about the life of a pirate. We bring you stories of adventure, to hoist the Jolly Roger and join us on a swashbuckling journey!</p>
</section>
<section id="crew">
<h2>Meet the Crew</h2>
<ul>
<li>Captain Blackbeard</li>
<li>First Mate Long John Silver</li>
<li>Navigator Anne Bonny</li>
<li>Gunner Black Bart Roberts</li>
</ul>
</section>
<section id="treasure">
<h2>Treasure Hunt</h2>
<p>Are you ready for a treasure hunt? Follow the clues to find the hidden treasure:</p>
<ol>
<li>Find the palm tree with the skull and crossbones.</li>
<li>Walk 10 paces due north.</li>
<li>Turn left and walk 5 paces.</li>
<li>Look for the big rock with a red X on it.</li>
<li>Dig beneath the rock to find the treasure!</li>
</ol>
</section>
</main>
<footer>
<p>&copy; 2023 Pirate Page</p>
</footer>
</body>
</html>
```

Figura #3

Después de realizar las respectivas pruebas de que esté funcionando el balancer, Si actualizamos la página o se presiona F5, observamos que el balanceo está funcionando correctamente ya que al entrar con la ip del proxy este nos redirige a las páginas index alojadas en los otros dos servidores. Ver Figura #4.

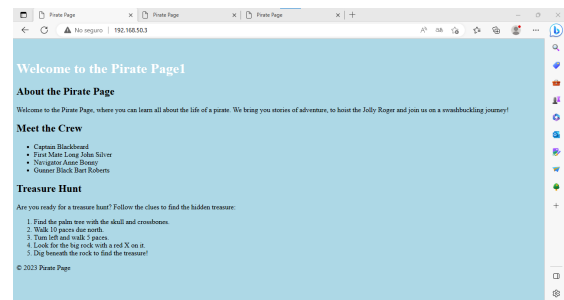
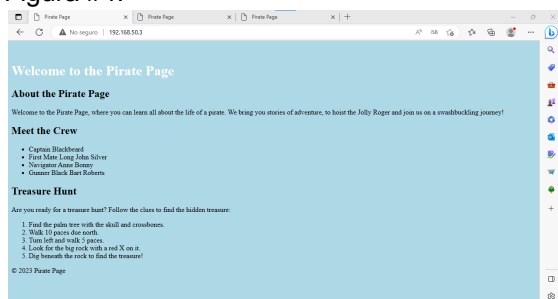


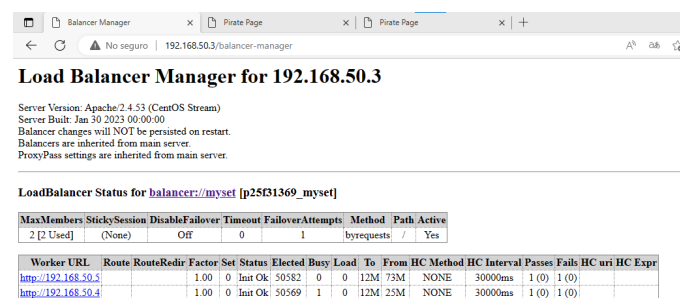
Figura #4

Seguimos configurando el Balance Manger creando nuevamente un archivo de configuración del manager en la ruta /etc/httpd/conf.d/. El archivo se debe llamar proxy.conf, dentro de este archivo de configuración debemos agregar lo siguiente. Ver figura #5

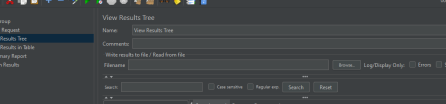
```
vagrant@proxy:/etc/httpd/conf.d
<IfModule mod_proxy.c>
ProxyRequests Off
<Proxy *>
Require all granted
</Proxy>
<Location "/balancer-manager">
SetHandler balancer-manager
Options Indexes FollowSymLinks
AllowOverride None
Require all granted
</Location>
<Proxy balancer://myset>
BalancerMember http://192.168.50.5
BalancerMember http://192.168.50.4
ProxySet lbmethod=byrequests
</Proxy>
ProxyPass /balancer-manager !
ProxyPass / balancer://myset/
ProxyPassReverse / balancer://myset/
</IfModule>
```

Figura #5

Una vez hecho toda la configuración logramos observar el panel de gestión del balanceo de carga además de los servidores, también observamos el estado, miembros y demás.



The screenshot shows the Burp Suite interface with the HTTP Request tab selected. The left sidebar displays a tree view of the project, including the HTTP Request tab. The main pane shows the details of the selected request, including the Name, Comment, Method, URL, Headers, Body, and Parameters. The URL is http://10.10.10.10:8080. The Parameters section is empty.




The screenshot shows the Splunk Enterprise console interface. On the left, the 'Search' sidebar is visible, showing a list of recent searches. The 'HITIP Request' search is selected. The main panel displays the 'View Results' interface for this search. The search bar at the top contains the query 'HITIP Request'. Below the search bar, the 'View Results' tab is active, showing a table of search results. The table has columns for 'Name', 'View Results Time', and 'Comments'. The search results are filtered to show only 'HITIP Request' events. The top of the console shows the search bar with the query 'HITIP Request' and the 'View Results' button.

The screenshot shows the AWS IAM console interface. On the left, the 'AWS-ReadOnlyAccess' policy is selected under the 'Policies' tab. The main area displays the 'View Results in Table' page for this policy. The table contains 16 rows of data, all representing 'AWS-ReadOnlyAccess' permissions with a status of 'Active'. The table columns are: Sample ID, Start Time, Threshold Name, IAM Role, Sample Type, Status, Batch, Start Date, Latency, and Count.

Sample ID	Start Time	Threshold Name	IAM Role	Sample Type	Status	Batch	Start Date	Latency	Count
1	2/16/2018 10:18	Threshold Group - AWS-ReadOnlyAccess	AWSTransientRole	AWSTransientRole	Active	1001	2/16/2018	258	258
2	2/16/2018 10:18	Threshold Group - AWS-ReadOnlyAccess	AWSTransientRole	AWSTransientRole	Active	1001	2/16/2018	258	258
3	2/16/2018 10:18	Threshold Group - AWS-ReadOnlyAccess	AWSTransientRole	AWSTransientRole	Active	1001	2/16/2018	258	258
4	2/16/2018 10:18	Threshold Group - AWS-ReadOnlyAccess	AWSTransientRole	AWSTransientRole	Active	1001	2/16/2018	258	258
5	2/16/2018 10:18	Threshold Group - AWS-ReadOnlyAccess	AWSTransientRole	AWSTransientRole	Active	1001	2/16/2018	258	258
6	2/16/2018 10:18	Threshold Group - AWS-ReadOnlyAccess	AWSTransientRole	AWSTransientRole	Active	1001	2/16/2018	258	258
7	2/16/2018 10:18	Threshold Group - AWS-ReadOnlyAccess	AWSTransientRole	AWSTransientRole	Active	1001	2/16/2018	258	258
8	2/16/2018 10:18	Threshold Group - AWS-ReadOnlyAccess	AWSTransientRole	AWSTransientRole	Active	1001	2/16/2018	258	258
9	2/16/2018 10:18	Threshold Group - AWS-ReadOnlyAccess	AWSTransientRole	AWSTransientRole	Active	1001	2/16/2018	258	258
10	2/16/2018 10:18	Threshold Group - AWS-ReadOnlyAccess	AWSTransientRole	AWSTransientRole	Active	1001	2/16/2018	258	258
11	2/16/2018 10:18	Threshold Group - AWS-ReadOnlyAccess	AWSTransientRole	AWSTransientRole	Active	1001	2/16/2018	258	258
12	2/16/2018 10:18	Threshold Group - AWS-ReadOnlyAccess	AWSTransientRole	AWSTransientRole	Active	1001	2/16/2018	258	258
13	2/16/2018 10:18	Threshold Group - AWS-ReadOnlyAccess	AWSTransientRole	AWSTransientRole	Active	1001	2/16/2018	258	258
14	2/16/2018 10:18	Threshold Group - AWS-ReadOnlyAccess	AWSTransientRole	AWSTransientRole	Active	1001	2/16/2018	258	258
15	2/16/2018 10:18	Threshold Group - AWS-ReadOnlyAccess	AWSTransientRole	AWSTransientRole	Active	1001	2/16/2018	258	258
16	2/16/2018 10:18	Threshold Group - AWS-ReadOnlyAccess	AWSTransientRole	AWSTransientRole	Active	1001	2/16/2018	258	258

Summary Report

Label	# Samples	Average	Min	Max	Std Dev	Error %	Throughput	ReducedMB/s	Sent MB/s	Avg. Rate
HDFS Request	1000	1462	7	8888	1959.77	0.00%	875.1 ms	1340.38	89.08	1332
GC/IO	1000	1462	7	8888	1959.77	0.00%	875.1 ms	1340.38	89.08	1332

[illegible]

```
Vagrantfile: Bloc de notas

Archivo  Edición  Formato  Ver  Ayuda

Vagrant.configure("2") do |config|
  config.vm.boot_timeout = 600
  if Vagrant.has_plugin? "vagrant-vbguest"
    config.ssh.username = 'vagrant'
    config.ssh.password = 'vagrant'
    config.ssh.insert_key = 'true'
    config.vbguest.no_install = true
    config.vbguest.auto_update = false
    config.vbguest.no_remote = true
  end

  config.vm.define :servidor1 do |servidor1|
    config.vm.box = "bento/centos-stream-9"
    servidor1.vm.network :private_network, ip: "192.168.50.5"
    servidor1.vm.hostname = "servidor1"
    servidor1.vm.provision "shell", path: "./config/server_conf.sh"
  end

  config.vm.define :servidor2 do |servidor2|
    config.vm.box = "bento/centos-stream-9"
    servidor2.vm.network :private_network, ip: "192.168.50.4"
    servidor2.vm.hostname = "servidor2"
    servidor2.vm.provision "shell", path: "./config/server_conf.sh"
  end

  config.vm.define :proxy do |proxy|
    config.vm.box = "bento/centos-stream-9"
    proxy.vm.network :private_network, ip: "192.168.50.3"
    proxy.vm.hostname = "proxy"
    proxy.vm.provision "shell", path: "./config/proxy_conf.sh"
  end
end
```

Línea 1, columna 1    100%    Windows (CRLF)    UTF-8

también se puede utilizar como equilibrador de carga.

- **HAProxy:** HAProxy es un servidor proxy y equilibrador de carga de código abierto y alto rendimiento que admite aplicaciones basadas en TCP y HTTP.

- **AWS Elastic Load Balancer:** AWS ELB es un servicio de balanceador de carga completamente administrado proporcionado por Amazon Web Services que puede distribuir el tráfico en varias instancias o zonas de disponibilidad.

Estos son solo algunos ejemplos de las soluciones de equilibrio de carga disponibles. Cada solución tiene su propio conjunto de características y beneficios, por lo que es importante evaluar sus requisitos específicos y elegir la solución que mejor se adapte a sus necesidades.

### ***VIII. Conclusiones***

- Mediante el balanceo de carga, podemos asignar de manera equitativa las solicitudes de los usuarios a las dos rutas o direcciones IP disponibles, en función del tráfico generado.
- La distribución del tráfico entrante a través del balanceo de carga proporciona un mejor rendimiento y una mayor capacidad para hacer frente a posibles fallos.
- En caso de que una de las rutas falle, la otra asumirá automáticamente todas las solicitudes.
- Si se incrementa el número de nodos disponibles, se mejorará tanto el rendimiento como la capacidad de tolerancia a fallos.

### ***IX. Bibliografía***

- [1] [https://httpd.apache.org/docs/current/howto/reverse\\_proxy.html](https://httpd.apache.org/docs/current/howto/reverse_proxy.html)
- [2] [https://www.digitalocean.com/community/tutorials/how-to-use-apache-as-a-reverse-proxy-with-mod\\_proxy-on-centos-7](https://www.digitalocean.com/community/tutorials/how-to-use-apache-as-a-reverse-proxy-with-mod_proxy-on-centos-7)
- [3] [https://www.server-world.info/en/note?os=CentOS\\_8&p=httpd&f=12](https://www.server-world.info/en/note?os=CentOS_8&p=httpd&f=12)
- [4] <https://www.centlinux.com/2019/01/configure-apache-http-load-balancer-centos-7.html>