Práctica 1.10.a

Implantación de Aplicaciones Web

José Juan Sánchez Hernández

Curso 2024/2025

Índice

1 Balanceador de carga con Apache **1**

1.1 ¿Qué es un balanceador de carga? 1
1.2 ¿Qué es un proxy inverso? 3
1.3 Activación de los módulos necesarios de Apache para configurarlo como proxy inverso 3
1.4 Configuración de la política de balanceo de carga 4
1.5 Configuración de Apache para trabajar como balanceador de carga para el tráfico HTTP 5
1.6 Configuración de Apache para trabajar como balanceador de carga para el tráfico HTTPS 6
1.7 Entregables 6
1.7.1 Documento técnico 7
1.7.2 Scripts de Bash 7

2 Referencias **8**

3 Licencia **9**

Índice de figuras

Índice de cuadros

1 Balanceador de carga con Apache

En esta práctica deberá automatizar la instalación y configuración de una aplicación web LAMP en **cuatro máquinas virtuales EC2** de Amazon Web Services (AWS), con la última versión de Ubuntu Server. En esta práctica vamos a usar una máquina virtual con Apache HTTP Server como un proxy inverso para hacer de balanceador de carga.

El objetivo de esta práctica es crear una arquitectura de **alta disponibilidad** que sea **escalable** y **redundante**, de modo que podamos balancear la carga entre todos los frontales web.

La arquitectura estará formada por:

- Un balanceador de carga, implementado con un Apache HTTP Server configurado como proxy inverso.
- Una capa de *front-end*, formada por dos servidores web con Apache HTTP Server.
- Una capa de *back-end*, formada por un servidor MySQL.

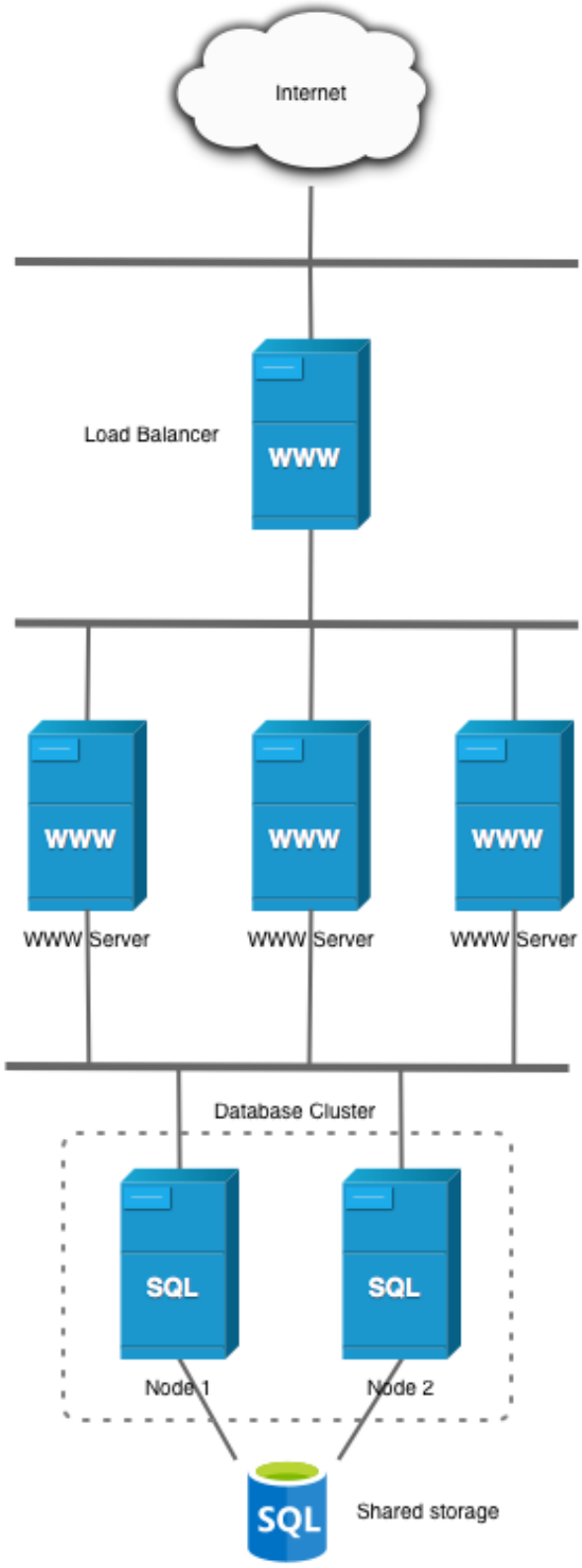
Necesitará crear cuatro máquinas virtuales:

- Balanceador.
- Frontal Web 1.
- Frontal Web 2.
- Servidor de Base de Datos MySQL.

1.1 ¿Qué es un balanceador de carga?

Un **balanceador de carga** es un dispositivo **hardware** o **software** que se pone al frente de un conjunto de servidores y se encarga de asignar o balancear las peticiones que llegan de los clientes hacia los servidores.

Estos dispositivos permiten distribuir el tráfico de red entre varios servidores o dispositivos de red, con el fin de mejorar el rendimiento y la disponibilidad de un sistema o aplicación.


Internet
Load Balancer
WWW
WWW
WWW
WWW
WWW Server
WWW Server
WWW Server
Database Cluster
SQL
SQL
Node 1
Node 2
SQL
Shared storage

Ejemplos de balanceadores de carga hardware:

- F5 BIG-IP.
- Kemp LoadMaster.

Ejemplos de balanceadores de carga software:

- HAProxy.
- Nginx.
- Apache HTTP Server.

Se recomienda la lectura de las siguientes referencias:

- What is Load Balancing?. DigitalOcean.
- Balanceador de carga. Wikipedia.

1.2 ¿Qué es un proxy inverso?

Un **proxy inverso** es un tipo de servidor proxy que hace de intermediario entre un cliente y uno o más servidores. El cliente realiza las peticiones a los servidores a través del proxy inverso y las respuestas de los servidores hacia el cliente también se envían a través del proxy inverso.

En esta práctica vamos a configurar un **servidor web Apache como proxy inverso** para que trabaje como balanceador de carga para el tráfico HTTP y HTTPS.

Se recomienda la lectura de la siguiente referencia:

- Proxy Inverso. Wikipedia.

1.3 Activación de los módulos necesarios de Apache para configurarlo como proxy inverso

Para configurar el servidor web Apache como proxy inverso tenemos que activar como mínimo los siguientes módulos:

```
1 sudo a2enmod proxy
2 sudo a2enmod proxy_http
3 sudo a2enmod proxy_balancer
```

A continuación, se muestra una breve descripción de cada uno de ellos:

- `proxy`: Permite configurar el servidor web como un proxy inverso.
- `proxy_http`: Permite configurar el servidor web como un proxy inverso para el protocolo HTTP (hasta la versión 1.1). Si nuestro servidor web está configurado para trabajar con la version 2.0 de HTTP, entonces tendremos que activar el módulo `proxy_http2`.

- `proxy_balancer`: Este módulo permite configurar cuál será la política de balanceo de carga que utilizará el servidor web.

Después de habilitar los módulos es necesario reiniciar el servicio de Apache.

```
sudo systemctl restart apache2
```

Otros módulos

También es posible activar otros módulos que nos permiten realizar otras tareas según nuestras necesidades:

- `proxy_http2`: Permite configurar el servidor web como un proxy inverso para el protocolo HTTP/2.
- `proxy_ajp`: Permite configurar el servidor web como un proxy inverso para el protocolo AJP (Apache JServ Protocol).
- `deflate`: Permite comprimir el contenido que se envía al cliente.
- `headers`: Permite al servidor web manipular las cabeceras de las peticiones/respuestas HTTP que envía/recibe.
- `proxy_connect`: Permite configurar el servidor web como un servidor proxy que puede establecer conexiones HTTPS con los servidores donde distribuye la carga, utilizando el método CONNECT de HTTP.
- `proxy_html`: Permite configurar el servidor web como un servidor proxy que puede filtrar y modificar el contenido HTML de las páginas web que se reciben de los servidores donde se distribuye la carga.

1.4 Configuración de la política de balanceo de carga

Para configurar la política de balanceo de carga es necesario tener activado previamente el módulo `proxy_balancer` con el siguiente comando:

```
sudo a2enmod proxy_balancer
```

El balanceo de carga que vamos a utilizar en esta práctica será un balanceo de tipo **Round Robin**. Este método de balanceo de carga consiste en distribuir las peticiones entre los servidores de forma secuencial, de forma que cada vez que llegue una nueva petición se envía al siguiente servidor de la lista de servidores configurados en el servidor Apache.

Para activar este método de balance tenemos que activar el módulo `lbmethod_byrequests`:

```
sudo a2enmod lbmethod_byrequests
```

Este método de balanceo también permite distribuir las peticiones entre los servidores en función de los parámetros `lbfactor` y `lbstatus`.

Puede consultar más información sobre este módulo en la documentación oficial.

Recuerde que después de habilitar los módulos es necesario reiniciar el servicio de Apache.

```
sudo systemctl restart apache2
```

Otros métodos de balanceo de carga

Los módulos disponibles para configurar el método de balanceo de carga son los siguientes:

- `lbmethod_bybusyness`: Este planificador de balanceo de carga tiene en cuenta el número de peticiones que tiene asignado cada servidor actualmente. Cuando llega una nueva petición al balanceador, se asigna al servidor que tenga menos peticiones activas. Puede consultar más información sobre este módulo en la documentación oficial.
- `lbmethod_bytraffic`: Este módulo es similar al anterior, pero en este caso el parámetro `lbfactor` representa la cantidad de tráfico en *bytes* que se le asigna a cada servidor. Por ejemplo, un servidor con un `lbfactor` de 2 recibirá el doble de bytes que otro con un `lbfactor` de 1. Puede consultar más información sobre este módulo en la documentación oficial.
- `lbmethod_heartbeat`: Este módulo realiza el balanceo de carga basándose en el estado de los servidores. Los servidores envían mensajes de *heartbeat* o latidos al balanceador cada cierto tiempo, si un servidor deja de enviar mensajes de *heartbeat* el balanceador considera que está caído y no le asigna más peticiones. Puede consultar más información sobre este módulo en la documentación oficial.

1.5 Configuración de Apache para trabajar como balanceador de carga para el tráfico HTTP

Paso 1. Creación de un nuevo archivo de VirtualHost

Creamos un nuevo archivo de configuración para crear un VirtualHost con la configuración del proxy inverso.

Creamos el archivo `load-balancer.conf` en el directorio `/etc/apache2/sites-available`:

```
sudo nano /etc/apache2/sites-available/load-balancer.conf
```

Le añadimos las directivas `Proxy` y `ProxyPass`.

```
<VirtualHost *:80>
    <Proxy balancer://mycluster>
        # Server 1
        BalancerMember http://IP_HTTP_SERVER_1

        # Server 2
        BalancerMember http://IP_HTTP_SERVER_2
    </Proxy>

    ProxyPass / balancer://mycluster/
</VirtualHost>
```

Tendremos que reemplazar `IP_HTTP_SERVER_1` y `IP_HTTP_SERVER_2` por las direcciones IPs de las dos máquinas que estamos utilizando como Front-End.

Paso 2. Habilitamos el VirtualHost que acabmos de crear

Habilitamos el VirtualHost que acabamos de crear con el siguiente comando:

```
sudo a2ensite load-balancer.conf
```

Deshabilitamos el VirtualHost que tiene Apache configurado por defecto:

```
sudo a2dissite 000-default.conf
```

Puede consultar los VirtualHost que tiene habilitados Apache con el comando:

```
sudo apache2ctl -S
```

Paso 3. Reiniciamos el servicio de Apache

Una vez aplicados los cambios reiniciamos el servicio de Apache:

```
sudo systemctl restart apache2
```

1.6 Configuración de Apache para trabajar como balanceador de carga para el tráfico HTTPS

Una vez que haya comprobado que el balanceo de carga del tráfico HTTP se realiza de forma correcta, puede empezar a configurar el balanceador de carga para el tráfico HTTPS.

Para poder habilitar el protocolo HTTPS en el balanceador es necesario obtener un **certificado de SSL/TLS**. Este certificado tiene que ser emitido por una **autoridad de certificación** (AC). En esta práctica vamos a utilizar Certbot para obtener un certificado de la Autoridad de Certificación Let's Encrypt.

Se recomienda la lectura de la práctica HTTPS con Let's Encrypt y Certbot para conocer cuáles son los pasos necesarios para obtener un certificado de Let's Encrypt con Certbot.

Los pasos que tendrá que realizar son los siguientes:

- **Crear una dirección IP elástica en AWS** y asociarla a la instancia EC2 que hace de balanceador.
- **Registrar un nombre de dominio** en algún proveedor de nombres de dominio gratuito. Por ejemplo, puede hacer uso de Freenom o No-IP.
- **Configurar los registros DNS del proveedor de nombres de dominio** para que el nombre de dominio de ha registrado pueda resolver hacia la dirección IP elástica que ha asociado con su instancia EC2 de AWS.
- **Instalar y configurar el cliente ACME Certbot** en su instancia EC2 de AWS que hace balanceador, siguiendo los pasos de la documentación oficial.

1.7 Entregables

Deberá crear un repositorio en GitHub con el nombre de la práctica y añadir al profesor como colaborador.

El repositorio debe tener el siguiente contenido:

- Un **documento técnico** con la descripción de todos los pasos que se han llevado a cabo.
- Los **scripts de Bash** que se han utilizado para automatizar la instalación y configuración de la aplicación web propuesta utilizando la arquitectura web de tres niveles.

Además del contenido anterior puede ser necesario crear otros archivos de configuración. A continuación se muestra un ejemplo de cómo puede ser la estructura del repositorio:

```
.
├── README.md
├── conf
│   ├── load-balancer.conf
│   └── 000-default.conf
├── scripts
│   ├── .env
│   ├── install_load_balancer.sh
│   ├── install_lamp_frontend.sh
│   ├── install_lamp_backend.sh
│   ├── setup_letsencrypt_https.sh
│   └── deploy.sh
```

1.7.1 Documento técnico

El documento técnico `README.md` tiene que estar escrito en Markdown y debe incluir **como mínimo** los siguientes contenidos:

- Descripción del proceso de instalación de la instalación de la aplicación web propuesta.

1.7.2 Scripts de Bash

El directorio `scripts` debe incluir los siguientes archivos:

- `.env`: Este archivo contiene todas las variables de configuración que se utilizarán en los scripts de Bash.
- `install_load_balancer.sh`: Script de Bash con la automatización del proceso de instalación del servidor web Apache como balanceador de carga.
- `install_lamp_frontend.sh`: Script de Bash con la automatización del proceso de instalación de la pila LAMP en las máquinas de frontend.
- `install_lamp_backend.sh`: Script de Bash con la automatización del proceso de instalación de la pila LAMP en la máquina de backend.
- `setup_letsencrypt_https.sh`: Script de Bash con la automatización del proceso de solicitar un certificado SSL/TLS de Let's Encrypt y configurarlo en el servidor web Apache que hace de balanceador de carga.
- `deploy`: Script de Bash con la automatización del proceso de instalación de la aplicación web propuesta.

2 Referencias

- Guía de proxy inverso para Apache HTTP Server
- How to use Apache HTTP Server as reverse-proxy using mod_proxy extension
- Load Balancing Techniques and Optimizations. Jason Potter.

3 Licencia

Esta página forma parte del curso Implantación de Aplicaciones Web de José Juan Sánchez Hernández y su contenido se distribuye bajo una licencia Creative Commons Reconocimiento-NoComercial-CompartirIgual 4.0 Internacional.