Práctica 8

Implantación de Aplicaciones Web

José Juan Sánchez Hernández

Curso 2023/2024

Índice

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.

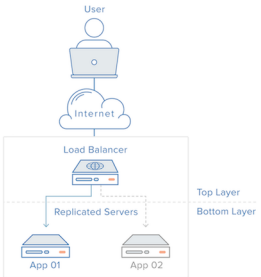
Necesitará crear cuatro máquinas virtuales:

- 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.



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?

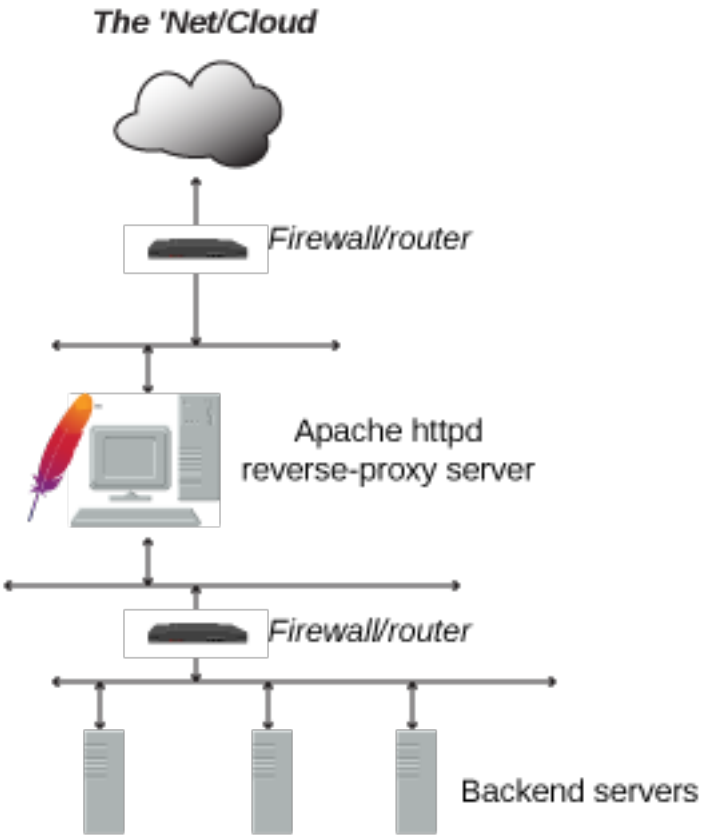
Un **proxy inverso** es un tipo de servidor proxy que hace de intermediario entre un cliente y uno o más servidores. El cliente realiza las peticiones a los servidores a través del proxy inverso y las respuestas de los servidores hacia el cliente también se envían a través del proxy inverso.

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 Arquitectura típica de proxy inverso con Apache



1.4 Activación de los módulos necesarios en Apache para configurarlo como proxy inverso

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

```
a2enmod proxy
a2enmod proxy_http
```

También es posible activar otros módulos que nos permite realizar otras tareas:

```
a2enmod proxy_ajp
a2enmod rewrite
a2enmod deflate
a2enmod headers
a2enmod proxy_connect
a2enmod proxy_html
a2enmod lbmethod_byrequests
```

A continuación se muestra una breve descripción de cada uno de los módulos que hemos activado:

- `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.
- `proxy_ajp`: Permite configurar el servidor web como un proxy inverso para el protocolo AJP (Apache JServ Protocol).
- `rewrite`: Permite al servidor reescribir las peticiones HTTP que recibe.
- `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.

1.5 Módulos de Apache para configurar el método de balanceo de carga

El balanceo de carga que se utiliza por defecto en el servidor Apache cuando se configura como proxy inverso es 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 configurar otros métodos de balanceo de carga es necesario tener activado previamente el módulo `proxy_balancer` con el siguiente comando:

```
1 a2enmod proxy_balancer
```

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.

  Para activar este módulo, ejecutamos el siguiente comando:

  ```
  1 a2enmod lbmethod_bybusyness
  ```

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

- `lbmethod_byrequests`: Este módulo añade un nuevo método de balanceo de carga que permite distribuir las peticiones entre los servidores en función de los parámetros `lbfactor` y `lbstatus` que se le pasan a la directiva `ProxySet`.

  Para activar este módulo, ejecutamos el siguiente comando:

```
a2enmod lbmethod_byrequests
```

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.

  Para activar este módulo, ejecutamos el siguiente comando:

  ```
  a2enmod lbmethod_bytraffic
  ```

  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.

  Para activar este módulo, ejecutamos el siguiente comando:

  ```
  a2enmod lbmethod_heartbeat
  ```

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

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

Editamos el archivo `000-default.conf` que está en el directorio `/etc/apache2/sites-available`:

```
sudo nano /etc/apache2/sites-available/000-default.conf
```

Añadimos las directivas `Proxy` y `ProxyPass` dentro de `VirtualHost`.

```
<VirtualHost *:80>
    # Dejamos la configuración del VirtualHost como estaba
    # sólo hay que añadir las siguiente directivas: Proxy y ProxyPass

    <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.

1.6.1 Reiniciamos el servicio de Apache

Una vez aplicados los cambios reiniciamos el servicio de Apache:

```
1 sudo systemctl restart apache2
```

1.7 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 seguridad**. 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.

1.8 Entregables

En esta práctica habrá que entregar un **documento técnico** con la descripción de los pasos que se han llevado a cabo.

El documento debe incluir **como mínimo** los siguientes contenidos:

- URL del repositorio de GitHub donde se ha alojado el documento técnico escrito en Markdown.
- *Scripts* de bash utilizados para realizar el aprovisionamiento de las máquinas virtuales.

- Tenga en cuenta que el aprovisionamiento de las máquinas virtuales se realizará mediante un *script* de *bash*. Cada máquina usará su propio *script*. El contenido de cada uno de los *scripts* deberá ser incluido en el documento y **deberá describir qué acciones se han ido realizando en cada uno de ellos**.
- URL del sitio web con HTTPS habilitado.

2 Créditos

Las imágenes utilizadas en esta guía se han obtenido de DigitalOcean y Apache.

3 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.

4 Licencia