MÓDULO 1(anexos):

Proxy inverso

1. ¿Qué es un proxy inverso? ¿Para qué se utiliza?

Un **proxy inverso** es un servidor intermediario que actúa como punto de entrada para los clientes que desean acceder a un servicio alojado en uno o varios servidores internos. En lugar de que los clientes se comuniquen directamente con los servidores backend, el proxy inverso recibe las solicitudes, las procesa y luego las reenvía al servidor apropiado.

Usos comunes de un proxy inverso:

- **Seguridad:** Oculta la infraestructura interna de los servidores backend, protegiendo contra ataques directos.
- **Caché:** Almacena respuestas frecuentes para reducir la carga en los servidores backend y mejorar el rendimiento.
- **Balanceo de carga:** Distribuye las solicitudes entre varios servidores para evitar sobrecargas.
- **Compresión:** Comprime datos antes de enviarlos al cliente, optimizando el ancho de banda.
- **SSL/TLS terminación:** Maneja las conexiones cifradas (HTTPS) y las convierte en HTTP para los servidores backend.
- **Alta disponibilidad:** Asegura que el servicio esté siempre disponible, incluso si un servidor backend falla.

---

2. ¿Qué es un balanceador de carga? ¿Para qué se usa?

Un **balanceador de carga** es un sistema que distribuye las solicitudes entrantes entre varios servidores backend para asegurar que ningún servidor esté sobrecargado. Esto mejora el rendimiento, la escalabilidad y la disponibilidad del servicio.

Usos comunes del balanceo de carga:

- **Escalabilidad horizontal:** Permite agregar más servidores para manejar un mayor volumen de tráfico.
- **Redundancia:** Si un servidor falla, el balanceador redirige el tráfico a otros servidores disponibles.
- **Optimización de recursos:** Asegura que todos los servidores sean utilizados de manera eficiente.

- **Mejora del rendimiento:** Reduce los tiempos de respuesta al distribuir la carga de manera equitativa.

---

3. Sistemas de balanceo de carga en Linux

En Linux, existen varias herramientas y tecnologías para implementar balanceadores de carga:

- **HAProxy:** Un balanceador de carga de alto rendimiento y proxy TCP/HTTP.
- **NGINX:** Un servidor web que también puede funcionar como proxy inverso y balanceador de carga.
- **Apache HTTP Server:** Puede configurarse como proxy inverso y balanceador de carga usando módulos como `mod_proxy` y `mod_proxy_balancer`.
- **Keepalived:** Se utiliza junto con HAProxy para proporcionar alta disponibilidad mediante IP flotantes.
- **Linux Virtual Server (LVS):** Una solución de balanceo de carga a nivel del kernel.

---

4. Algoritmos de balanceo de carga

Los algoritmos de balanceo de carga determinan cómo se distribuyen las solicitudes entre los servidores backend. Algunos de los más comunes son:

- **Round Robin:** Distribuye las solicitudes secuencialmente entre los servidores backend.
- **Least Connections:** Envía la solicitud al servidor con menos conexiones activas.
- **IP Hash:** Asigna un cliente a un servidor específico basándose en su dirección IP.
- **Weighted Round Robin:** Similar al Round Robin, pero permite asignar pesos a los servidores según su capacidad.
- **Random:** Distribuye las solicitudes aleatoriamente entre los servidores.
- **Least Response Time:** Envía la solicitud al servidor que responde más rápido.

---

5. Apache como proxy inverso

Apache puede configurarse como proxy inverso utilizando los módulos `mod_proxy` y `mod_proxy_balancer`. A continuación, se muestra un ejemplo de configuración:

Fichero de configuración (`httpd.conf`) explicado línea por línea:

```
# Habilitar los módulos necesarios
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

# Configurar el proxy inverso
<VirtualHost *:80>
ServerName example.com

# Definir el grupo de servidores backend
<Proxy balancer://mycluster>
BalancerMember http://backend1.example.com
BalancerMember http://backend2.example.com
ProxySet lbmethod=byrequests # Algoritmo de balanceo (Round Robin)
</Proxy>

# Redirigir todas las solicitudes al balanceador
ProxyPass / balancer://mycluster/
ProxyPassReverse / balancer://mycluster/

# Opcional: Configurar caché
CacheEnable disk /
CacheRoot "/var/cache/apache2"
</VirtualHost>
```

Explicación:

1. **`LoadModule`:** Carga los módulos necesarios para el proxy inverso y el balanceo de carga.
2. **`<Proxy>`:** Define un grupo de servidores backend (`balancer://mycluster`).
3. **`BalancerMember`:** Especifica los servidores backend.
4. **`ProxySet`:** Define el algoritmo de balanceo (`lbmethod=byrequests`).
5. **`ProxyPass` y `ProxyPassReverse`:** Redirigen las solicitudes al balanceador y ajustan las URLs de respuesta.
6. **Caché (opcional):** Almacena respuestas frecuentes para mejorar el rendimiento.

---

6. NGINX como proxy inverso

NGINX es una alternativa popular para configurar un proxy inverso y balanceador de carga. A continuación, un ejemplo de configuración:

Fichero de configuración (`nginx.conf`) explicado línea por línea:

```
http {
```

```
upstream mycluster {
server backend1.example.com;
server backend2.example.com;
# Algoritmo de balanceo (Round Robin por defecto)
}

server {
listen 80;
server_name example.com;

location / {
proxy_pass http://mycluster; # Redirigir al grupo de servidores
proxy_set_header Host $host;
proxy_set_header X-Real-IP $remote_addr;
proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
}
}
}
```

Explicación:

1. **`upstream`:** Define un grupo de servidores backend (`mycluster`).
2. **`server`:** Especifica los servidores backend.
3. **`location /`:** Define cómo manejar las solicitudes.
4. **`proxy_pass`:** Redirige las solicitudes al grupo de servidores.
5. **`proxy_set_header`:** Ajusta las cabeceras de las solicitudes para preservar información del cliente.

---

7. Instalación y configuración de HAProxy

HAProxy es una herramienta especializada en balanceo de carga y proxy inverso.

Instalación en Linux:

```
sudo apt update
sudo apt install haproxy
```

Configuración (`/etc/haproxy/haproxy.cfg`):

```
global
```

```
log /dev/log local0
log /dev/log local1 notice
chroot /var/lib/haproxy
stats socket /run/haproxy/admin.sock mode 660 level admin expose-fd listeners
stats timeout 30s
user haproxy
group haproxy
daemon

defaults
log global
mode http
option httplog
option dontlognull
timeout connect 5000ms
timeout client 50000ms
timeout server 50000ms

frontend http_front
bind *:80
default_backend http_back

backend http_back
balance roundrobin
server backend1 backend1.example.com:80 check
server backend2 backend2.example.com:80 check
```

Explicación:

1. **`global`:** Configuración global de HAProxy.
2. **`defaults`:** Configuración predeterminada para todos los frontends y backends.
3. **`frontend`:** Define el puerto de escucha y el backend predeterminado.
4. **`backend`:** Define los servidores backend y el algoritmo de balanceo (`roundrobin`).

Reiniciar HAProxy:

```
sudo systemctl restart haproxy
```