Integración HAProxy – Docker Swarm

1. Introducción

En los últimos años, la adopción de **arquitecturas basadas en microservicios** ha transformado la manera en que se diseñan y despliegan aplicaciones. Este enfoque busca mejorar la escalabilidad, resiliencia y facilidad de actualización de los sistemas, al descomponer las aplicaciones en servicios independientes que interactúan entre sí.

Para gestionar estas arquitecturas distribuidas, surgen plataformas de **orquestación de contenedores** que permiten automatizar la administración de múltiples instancias, su comunicación y escalado dinámico. **Docker Swarm**, el orquestador nativo de Docker, facilita la gestión de clústeres de contenedores ofreciendo simplicidad y una integración estrecha con el ecosistema Docker.

Sin embargo, por sí solo, Swarm ofrece un balanceo de carga básico que puede resultar limitado en escenarios productivos de alta demanda. En este punto, entra en juego **HAProxy**, un software de balanceo de carga ampliamente utilizado en la industria por su robustez, flexibilidad y soporte avanzado de protocolos, monitoreo y terminación TLS.

La integración de **HAProxy con Docker Swarm** permite alcanzar una infraestructura distribuida, escalable y tolerante a fallos, mejorando el control sobre el enrutamiento de tráfico y proporcionando un único punto de entrada seguro y eficiente.

2. Marco Teórico

2.1 Docker Swarm

Docker Swarm es el **orquestador nativo de Docker**, diseñado para transformar un conjunto de nodos (máquinas físicas o virtuales con Docker instalado) en un único clúster lógico. A través de este, los administradores pueden desplegar y gestionar servicios de manera distribuida sin necesidad de herramientas adicionales.

Características principales:

- **Escalabilidad:** permite aumentar o reducir el número de réplicas de un servicio según la demanda.
- **Alta disponibilidad:** distribuye contenedores en distintos nodos, garantizando continuidad del servicio en caso de fallos.
- **Overlay networks:** posibilita la comunicación segura y aislada entre servicios que se ejecutan en diferentes nodos.

- **Service discovery:** cada servicio es accesible mediante un nombre DNS interno, facilitando la interconexión de microservicios.

2.2 HAProxy

HAProxy (High Availability Proxy) es un software de código abierto especializado en **balanceo de carga y proxy inverso**, utilizado en entornos de alto rendimiento como bancos, aerolíneas y plataformas de comercio electrónico.

Funcionalidades clave:

- Algoritmos de balanceo como *round robin*, *least connections* y *source hashing*.
- Terminación **SSL/TLS**, centralizando la gestión de certificados.
- **Health checks** para detectar servicios en buen estado y descartar los fallidos.
- Integración con sistemas de monitoreo a través de métricas y endpoints de estadísticas.

2.3 Justificación de la Integración

El balanceador interno de Docker Swarm, si bien es funcional, es **limitado** y no ofrece un control detallado sobre el tráfico entrante.

Al integrar HAProxy con Swarm se obtienen beneficios como:

- Políticas avanzadas de enrutamiento (por dominio, cabeceras, rutas).
- Centralización de certificados SSL.
- Métricas y observabilidad en tiempo real.
- Failover rápido y reducción de riesgos de indisponibilidad.

2.4 Keepalived

En escenarios de producción, un único balanceador de carga puede convertirse en un **punto único de falla**. Para evitarlo, se emplea **Keepalived**, un software basado en el protocolo VRRP (*Virtual Router Redundancy Protocol*), que permite alta disponibilidad en balanceadores de carga.

Características principales:

- Failover automático entre nodos balanceadores.

- Direcciones IP virtuales flotantes para mantener un único punto de entrada.
- Integración transparente con HAProxy o Nginx.
- Monitoreo del estado de los servicios.

Gracias a Keepalived, se asegura que si el balanceador principal (MASTER) falla, el secundario (BACKUP) asuma el control sin que los clientes perciban interrupciones.

3. Implementación Técnica

3.1 Definición de un servicio en Docker Swarm

Ejemplo de despliegue de un servicio API con 3 réplicas:

```
docker service create \
  --name api-service \
  --replicas 3 \
  --publish 8080 \
  api-image:latest
```

3.2 Configuración de HAProxy

Archivo **haproxy.cfg**:

```
global
    log stdout format raw local0

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

frontend http_front
    bind *:80
    default_backend app_back

backend app_back
    balance roundrobin
```

```
    server api1 api-service:8080 check
```

3.3 Despliegue en Swarm con docker-compose

Archivo **docker-compose.yml**:

```
version: '3.8'

services:
  haproxy:
    image: haproxy:latest
    ports:
      - "80:80"
    volumes:
      - ./haproxy.cfg:/usr/local/etc/haproxy/haproxy.cfg:ro
    networks:
      - swarm-net
    deploy:
      replicas: 1
      placement:
        constraints: [node.role == manager]

  api-service:
    image: my-api-image:latest
    networks:
      - swarm-net
    deploy:
      replicas: 3

networks:
  swarm-net:
    driver: overlay
```

3.4 Monitoreo

- HAProxy puede exponer métricas vía socket de estadísticas o integrarse con **Prometheus**.

- Los datos recolectados pueden visualizarse en **Grafana**, lo que facilita el diagnóstico y análisis de rendimiento.

3.5 Configuración de Keepalived

Se configuran dos nodos (MASTER y BACKUP) con un **IP virtual compartido**.

Ejemplo de archivo en el nodo **MASTER**:

```
vrrp_instance VI_1 {
    state MASTER
    interface eth0
    virtual_router_id 51
    priority 100
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass 1234
    }
    virtual_ipaddress {
        192.168.1.200
    }
}
```

En el nodo **BACKUP**, la prioridad es menor (90).

3.6 Script de instalación de Keepalived

Ejemplo:

```
#!/bin/bash
sudo apt update
sudo apt install keepalived -y

echo "¿Este es el nodo ACTIVO? (s/n): "
read nodo

if [ "$nodo" == "s" ] || [ "$nodo" == "S" ]; then
    sudo cp keepalived.conf /etc/keepalived/keepalived.conf
    echo "Configuración nodo ACTIVO aplicada."
else
    sudo cp keepalived-backup.conf /etc/keepalived/keepalived.conf
    echo "Configuración nodo PASIVO aplicada."
fi

sudo systemctl enable keepalived
```

```
sudo systemctl restart keepalived
echo "Keepalived instalado y configurado."
```

4. Casos de Uso

- **Microservicios distribuidos** con múltiples réplicas que requieren balanceo inteligente.
- **Enrutamiento por dominios (vhosts)** en aplicaciones multi-cliente.
- **TLS centralizado**, reduciendo la complejidad en cada microservicio.
- **Alta disponibilidad** en balanceadores de carga críticos.
- **Infraestructuras sensibles** (banca, e-commerce, aerolíneas) donde la tolerancia a fallos es prioritaria.

5. Ventajas y Desafíos

Ventajas:

- Flexibilidad en el balanceo y enrutamiento.
- Integración nativa con las redes overlay de Swarm.
- Escalabilidad transparente de servicios.
- Métricas integradas para monitoreo y diagnóstico.
- Con Keepalived: redundancia real y failover automático.

Desafíos:

- La complejidad aumenta en escenarios grandes.
- Un único HAProxy sigue siendo un **single point of failure** sin Keepalived.
- Requiere un plan sólido de observabilidad y monitoreo.
- Necesidad de sincronizar configuraciones entre balanceadores.