1. Introducción

Con la creciente adopción de arquitecturas basadas en microservicios, surge la necesidad de plataformas que permitan orquestar contenedores y balancear la carga de manera eficiente. **Docker Swarm**, el orquestador nativo de Docker, facilita la gestión de clústeres de contenedores, mientras que **HAProxy** se ha consolidado como una de las soluciones más robustas y flexibles para el balanceo de carga y la terminación TLS.

La integración de ambas tecnologías permite desplegar aplicaciones distribuidas de manera escalable, con alta disponibilidad y balanceo de tráfico inteligente.

2. Marco Teórico

2.1 Docker Swarm

- Es el orquestador nativo de Docker.
- Permite administrar múltiples nodos Docker como un único clúster lógico.
- Principales características:
    - **Escalabilidad**: aumento o reducción de réplicas de servicios.
    - **Alta disponibilidad**: distribución automática de contenedores en diferentes nodos.
    - **Overlay networks**: redes virtuales que permiten comunicación segura entre servicios.
    - **Service Discovery**: resolución automática de DNS entre servicios.

2.2 HAProxy

- Software de código abierto especializado en **balanceo de carga** y **proxy inverso**.
- Permite manejar tráfico **TCP** y **HTTP** con configuraciones avanzadas.
- Funcionalidades clave:
    - Balanceo **round robin, least connections, source hashing**.
    - Terminación SSL/TLS.
    - Health checks para detección de servicios activos.
    - Soporte para métricas y monitoreo.

2.3 Justificación de la Integración

- **Swarm** no ofrece un balanceador de carga avanzado por defecto (usa un enrutador interno básico).
- **HAProxy** complementa a Swarm proporcionando:
    - Mayor control en políticas de enrutamiento.
    - Mejor visibilidad y meticas.
    - Manejo de SSL centralizado.
    - Failover rápido entre servicios.

2.4 Keepalived

Keepalived es un software de alta disponibilidad basado en el protocolo VRRP (Virtual Router Redundancy Protocol).

Su función principal es evitar que un balanceador de carga se convierta en un punto único de falla, permitiendo que haya un balanceador activo y otro de respaldo.

Características principales:

- Monitorización de servicios (health checks).
- Failover automático entre nodos balanceadores.
- Uso de direcciones IP virtuales flotantes.
- Integración común con balanceadores como HAProxy o Nginx.

En este proyecto, Keepalived se utilizará para que, si el nodo HAProxy principal falla, el nodo de respaldo tome el control y continúe repartiendo tráfico sin que los clientes lo noten.

3. Implementación Técnica

3.1 Definición de un Servicio en Docker Swarm

```
docker service create \

 --name api-service \

 --replicas 3 \
```

```
 --publish 8080 \

 api-image:latest
```

3.2 Configuración de HAProxy (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
```

- **frontend**: escucha en el puerto 80.
- **backend**: redirige tráfico a los contenedores del servicio api-service.
- La resolución api-service la maneja el DNS interno de Swarm.

3.3 Despliegue de HAProxy en Swarm

Archivo docker-compose.yml para Swarm:

```
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 expone estadísticas vía web (stats socket) o Prometheus exporter.
- Se pueden graficar métricas en **Grafana**.

3.5 Configuración de keepalived

Se definen al menos dos nodos balanceadores con HAProxy instalado.

- Nodo principal (MASTER)
- Nodo de respaldo (BACKUP)

Cada nodo tendrá un archivo de configuración distinto:

keepalived.conf (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
        }
}
```

keepalived-backup.conf (nodo BACKUP):

```
vrrp_instance VI_1 {
        state BACKUP
        interface eth0
        virtual_router_id 51
        priority 90
        advert_int 1
        authentication {
                auth_type PASS
                auth_pass 1234
        }
        virtual_ipaddress {
                192.168.1.200
        }
}
```

3.6 Script de instalación automática

Para simplificar la configuración, se preparó un script (`install_keepalived.sh`) que instala y configura Keepalived en cada nodo. El usuario solo debe indicar si el nodo será activo (MASTER) o pasivo (BACKUP):

```
#!/bin/bash
# Actualizar paquetes e instalar keepalived
sudo apt update
sudo apt install keepalived -y

# Detectar si es nodo activo o pasivo
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

# Habilitar y arrancar el servicio
sudo systemctl enable keepalived
sudo systemctl restart keepalived

echo "Keepalived instalado y configurado."
```

4. Casos de Uso

1. Microservicios con múltiples réplicas que requieren balanceo avanzado.
2. Escenarios multicliente, donde se necesita enrutamiento basado en dominios (vhosts).
3. TLS centralizado, para manejar certificados en un único punto de entrada.
4. Alta disponibilidad con varias instancias de HAProxy en modo activo-activo.
5. Escenarios donde se requiere tolerancia a fallos del balanceador de carga.
6. Infraestructuras críticas (banca, e-commerce, aerolíneas) donde el servicio no puede caerse aunque falle un nodo balanceador.

5. Ventajas y Desafíos

Ventajas

- Flexibilidad en balanceo de carga.
- Integración transparente con redes overlay de Swarm.
- Capacidad de escalar servicios sin modificar el balanceador.
- Métricas y monitoreo integrados.

Ventajas adicionales con Keepalived:

- Alta disponibilidad real, incluso si un balanceador HAProxy falla.
- Failover automático y transparente para el usuario.
- Se mantiene una única IP virtual como punto de entrada.

Desafíos

- La configuración puede volverse compleja en escenarios grandes.
- Un único HAProxy puede ser un punto único de fallo (se recomienda usar HAProxy en modo redundante con Keepalived o similares).
- Requiere un plan de observabilidad robusto.
- Requiere configurar varias máquinas balanceadoras.
- Necesidad de sincronizar configuraciones entre los HAProxy.