

HAProxy: del descubrimiento a la experiencia

Guía Completa y Soporte de Curso

Domine HAProxy, el equilibrador de carga de referencia. Aprenda a configurar la alta disponibilidad, la terminación SSL, las ACL y la optimización del rendimiento.

Categoría : Proxy

Documento generado el: 30/10/2026

Formadores

Martin LEKPA

Puesto : Tech Lead et formateur Observabilité

Site : <https://martin.lekpa.fr>

Email : martin@lekpa.fr

Para ir más lejos: nuestros cursos Proxy

Traefik

[Ver la formación en línea](#)

Preguntas Frecuentes	3
Contenido teórico (Artículos)	3
Programa Práctico (Videos y TPs)	4
HAProxy: del descubrimiento a la experiencia (Trabajos Prácticos)	5
Introducción al Equilibrio de Carga (Trabajos Prácticos)	6
El ecosistema HAProxy: de Open Source a HAProxy One (Trabajos Prácticos)	7
Instalación y Gestión del servicio (Trabajos Prácticos)	8
Anatomía de la configuración: Global, Frontend, Backend y Defaults (Trabajos Prácticos)	9
Guía detallada de directivas de configuración (Trabajos Prácticos)	10
Guía rápida de HAProxy: comandos esenciales (Trabajos Prácticos)	11
Algoritmos: Round Robin, Leastconn y Hash (Trabajos Prácticos)	12
Persistencia de sesión (stickiness) (Trabajos Prácticos)	13
Health checks y alta disponibilidad (Trabajos Prácticos)	14
Dominar las ACLs (Trabajos Prácticos)	15
Las ACL: la guía completa (Trabajos Prácticos)	16
Terminación SSL y Offloading (Trabajos Prácticos)	17
ACME: SSL/TLS automatizado (Trabajos Prácticos)	18
Certificados con Let's Encrypt (Trabajos Prácticos)	19
El Dashboard de Estadísticas (Trabajos Prácticos)	20
Alta y baja dinámica de backends (Trabajos Prácticos)	21
Visualizar la infraestructura: Topology Visualizer (Trabajos Prácticos)	22
HAProxy y Docker: construir un clúster balanceado de alta disponibilidad (Trabajos Prácticos)	23
Puesta en marcha de la infraestructura	23
Primeros pasos con la configuración	23
Gestión de los Frontends	23
Gestión de los Backends	23
Enrutamiento avanzado con ACL	23
Seguridad y endurecimiento	23
Observabilidad	23
Práctica: persistencia, health checks y página de estadísticas (Trabajos Prácticos)	37
Backend resiliente	37
Práctica: terminación SSL y enrutamiento por ACL (Trabajos Prácticos)	39
HTTPS y ACL	39
HAProxy Ingress Controller para Kubernetes (Trabajos Prácticos)	41
Ingress Controller: guía de producción (Trabajos Prácticos)	42
Buenas prácticas de HAProxy (Trabajos Prácticos)	43
Novedades de HAProxy: OpenTelemetry, JWT, rendimiento (Trabajos Prácticos)	44
Cuestionario (QCM)	45
Para ir más lejos: nuestros cursos Proxy	79

Preguntas Frecuentes

¿Por qué elegir HAProxy en lugar de otro proxy?

HAProxy es famoso por su extrema velocidad y fiabilidad. A diferencia de Nginx, que es un servidor web polivalente, HAProxy es un puro especialista en balanceo de carga, con funciones de comprobación de salud y de persistencia mucho más precisas.

¿HAProxy gestiona el tráfico HTTPS?

Sí, perfectamente. Puede hacer 'SSL Pass-through' (dejar que el backend descifre) o 'Terminación SSL' (HAProxy descifra y descarga a los servidores web).

¿Qué es una ACL en HAProxy?

Una ACL (Access Control List) permite tomar decisiones de enrutamiento inteligentes basadas en criterios como la URL, la dirección IP de origen o los encabezados HTTP.

Detalles del Programa Práctico

[HAProxy: del descubrimiento a la experiencia](#) (Trabajos Prácticos)

Ponga a prueba sus conocimientos con este cuestionario

Introducción al Equilibrio de Carga (Trabajos Prácticos)

Comprender los retos de la alta disponibilidad y el posicionamiento de HAProxy.

[El ecosistema HAProxy: de Open Source a HAProxy One](#) (Trabajos Prácticos)

El ecosistema HAProxy: de Open Source a HAProxy One

Instalación y Gestión del servicio (Trabajos Prácticos)

Instalar HAProxy en Linux y dominar los comandos de recarga en caliente.

[Anatomía de la configuración: Global, Frontend, Backend y Defaults](#) (Trabajos Prácticos)

Estudio anatómico del fichero de configuración haproxy.cfg.

[Guía detallada de directivas de configuración](#) (Trabajos Prácticos)

Guía detallada de directivas de configuración

[Guía rápida de HAProxy: comandos esenciales](#) (Trabajos Prácticos)

Guía rápida de HAProxy: comandos esenciales

[Algoritmos: Round Robin, Leastconn y Hash](#) (Trabajos Prácticos)

Cómo HAProxy elige el servidor de destino.

Persistencia de sesión (stickiness) (Trabajos Prácticos)

Persistencia de sesión (stickiness)

[Health checks y alta disponibilidad](#) (Trabajos Prácticos)

Health checks y alta disponibilidad

Dominar las ACLs (Trabajos Prácticos)

Enrutamiento basado en el path, el host o las cookies.

[Las ACL: la guía completa](#) (Trabajos Prácticos)

Las ACL: la guía completa

[Terminación SSL y Offloading](#) (Trabajos Prácticos)

Centralice sus certificados SSL en HAProxy.

[ACME: SSL/TLS automatizado](#) (Trabajos Prácticos)

ACME: SSL/TLS automatizado

[Certificados con Let's Encrypt](#) (Trabajos Prácticos)

Certificados con Let's Encrypt

[El Dashboard de Estadísticas](#) (Trabajos Prácticos)

Activar y asegurar la interfaz web de monitorización.

[Alta y baja dinámica de backends](#) (Trabajos Prácticos)

Alta y baja dinámica de backends

[Visualizar la infraestructura: Topology Visualizer](#) (Trabajos Prácticos)

Visualizar la infraestructura: Topology Visualizer

HAProxy y Docker: construir un clúster balanceado de alta disponibilidad (Trabajos Prácticos)

Puesta en marcha de una infraestructura completa con balanceo de carga y monitorización.

Puesta en marcha de la infraestructura

Despliegue de 3 servidores web, un servidor Nginx y un clúster HAProxy con Docker Compose.

IMPORTANTE: Debes leer [este artículo](#) antes de empezar este TP para dominar bien los fundamentos.

También te recomendamos seguir estas formaciones:

- [Docker](#): para entender mejor la importancia de la contenedorización
- [Prometheus](#): para entender mejor la noción de métricas y su gestión
- [Opentelemetry](#): para entender mejor la noción de métricas y su gestión
- [Grafana](#): para entender mejor la visualización de los datos de telemetría

. Pero no es imprescindible, porque en este TP te daremos todos los comandos para estos productos.

Obtención de las fuentes

Clona las fuentes del proyecto desde este repositorio <https://github.com/rousselTM/haproxy-formation.git> en la carpeta que elijas.

Corrección :

```
git clone https://github.com/rousselTM/haproxy-formation.git
```

Puesta en marcha de la infraestructura

Para probar nuestras configuraciones, necesitamos una infraestructura mínima. Ejecuta los siguientes comandos para desplegarla en Docker:

```
cd haproxy-formation
sudo docker compose -f compose-infra.yaml up -d
```

Atención: Todos los comandos docker compose deben ejecutarse en la carpeta haproxy-formation.

Primeros pasos con la configuración

Exploración de la estructura de la configuración y puesta en marcha de la modularidad.

Acceder al archivo base haproxy.cfg

El corazón de HAProxy está en su archivo de configuración principal: haproxy.cfg. En nuestro entorno Docker, este archivo está montado como volumen.

El archivo se encuentra en config/haproxy/haproxy.cfg. Puedes abrirlo con tu editor de texto habitual para observar las secciones ya presentes.

Arranque de la instancia base

Para probar nuestras configuraciones, necesitamos una infraestructura **HAProxy 3.4** con una configuración básica que modificaremos más adelante. Primero, vamos a comprobar que funciona con la configuración básica mediante el siguiente comando:

```
sudo docker compose up -d
```

Verificación del arranque

El objetivo es aprender a identificar errores de configuración a través de los logs del sistema y de Docker, y verificar que los logs generados por HAProxy no contienen errores.

Instalación en máquina virtual o física: los logs suelen estar disponibles en /var/log/haproxy.log o mediante el comando

```
journalctl -u haproxy
```

Instalación con Docker Compose (nuestro caso): Usa el siguiente comando para consultar los logs de todas las instancias:

```
sudo docker compose logs -f haproxy
```

o este comando para consultar los de una instancia concreta (nombre del contenedor 3: haproxy-formation-haproxy-3)

```
sudo docker logs <container_name>
```

Con la configuración proporcionada, puedes encontrarte errores como:

```
haproxy-1 | [ALERT] (1) : config : parsing [/usr/local/etc/haproxy/haproxy.cfg:71]: Missing LF on last line, file might have been truncated at position 32.
```

Debes entender el mensaje y corregirlo.

Corrección :

El error simplemente te indica que falta un salto de línea (LF) al final del archivo.

BUENA PRÁCTICA: Prueba siempre tus configuraciones antes de reiniciar tu instancia en producción: opción -c

La importancia de la modularidad

A gran escala, mantener un archivo haproxy.cfg monolítico se vuelve complejo y arriesgado. La modularidad permite separar las responsabilidades (p. ej.: un archivo por aplicación, uno para la monitorización, uno para la seguridad, etc.), lo que facilita el despliegue continuo y las revisiones de código, y limita los errores humanos durante las actualizaciones.

Por defecto, el servicio HAProxy se lanza con la opción -f haproxy.cfg. Puedes añadir tantas opciones -f

como archivos tengas, pero también puedes usar la misma opción pasando una carpeta como parámetro: HAProxy cargará todos los archivos que terminen en .cfg de esa carpeta.

¡Ten muy en cuenta la dependencia entre los módulos!

En nuestro caso, con nuestra configuración de Docker Compose, usa el siguiente comando para pasar a usar la carpeta conf.d (acuérdate de mover el archivo de configuración a esta carpeta), que está al mismo nivel que el archivo haproxy.cfg. Los bloques de configuración de los ejercicios siguientes deberán colocarse, por tanto, en un archivo .cfg de esta carpeta, por ejemplo config/haproxy/conf.d/tp.cfg:

```
sudo docker compose down; export HAPROXY_CONFIG=conf.d; sudo -E docker compose up -d
```

Recarga de la configuración

Cada modificación del archivo debe ser tomada en cuenta por el motor de HAProxy.

Reload vs Restart: En producción, no uses nunca restart, porque corta las conexiones. El **Reload** (señal HUP) permite cargar la nueva configuración sin ninguna interrupción del servicio.

Debes encontrar el comando para recargar tu servidor sin cortar las conexiones en curso, con systemctl y con Docker.

Corrección :

Para recargar con systemctl:

```
sudo systemctl reload haproxy
```

Para recargar limpiamente con Docker Compose (Reload):

```
sudo docker compose kill -s HUP haproxy
```

Para un reinicio completo (Restart):

```
sudo docker compose restart haproxy
```

Para saber más, consulta el artículo sobre las [buenas prácticas de configuración](#).

Gestión de los Frontends

Configuración de los puntos de entrada para las estadísticas y el tráfico HTTP.

Punto de entrada para la administración

Crea un punto de entrada dedicado a la monitorización llamado 'admin'. Este punto de entrada debe escuchar en el puerto 8404 (puerto por defecto). Debes activar la interfaz de estadísticas, definir la raíz (/) como URI de acceso y configurar una actualización automática cada 5 segundos. Ten en cuenta que puedes hacer esta configuración con un bloque frontend o con un bloque listen; el resultado será idéntico.

Como hemos pasado a cargar la carpeta conf.d, añade esta configuración en un archivo .cfg de esa carpeta. Para aplicar tus cambios, usa el comando `export HAPROXY_CONFIG=conf.d; sudo -E docker compose restart haproxy` (en producción, haz siempre un reload).

Corrección :

Puedes editar el archivo de configuración de HAProxy para añadir esta configuración:

```
frontend admin
  bind :8404
  stats enable
  stats uri /
  stats refresh 5s
```

Y después, reiniciar el servicio.

Prueba del acceso a las estadísticas

Después de configurar el frontend 'admin', verifica que la interfaz de monitorización funciona correctamente. Puedes usar tu navegador o un comando de terminal para probar la accesibilidad del servicio en su puerto dedicado.

Corrección :

Verifica el acceso abriendo la siguiente URL en tu navegador:

```
http://localhost:8404
```

También puedes validar la conectividad con curl:

```
curl http://localhost:8404
```

Frontend HTTP

Configura el frontend principal llamado 'http_in' para recibir el tráfico web de los usuarios. Debe escuchar en el puerto 80 y enviar por defecto todas las peticiones al grupo de servidores 'default' (que se creará más adelante).

Corrección :

```
frontend http_in
  bind :80
  default_backend default
```

[Gestión de los Backends](#)

Configuración de los servidores de destino que procesan las peticiones.

El Backend por defecto

Crea un grupo de servidores llamado 'default'. Este backend se usará para procesar las peticiones que no coincidan con ninguna regla específica. Debe funcionar en modo HTTP, usar el algoritmo 'roundrobin' y apuntar al servidor 'web1'. Configura una regla para que este backend sirva siempre el archivo 'default.html'.

Corrección :

```
backend default
  mode http
  balance roundrobin
  # Reescritura de la ruta para mostrar la página de mantenimiento
```

```
http-request set-path /default.html
# Declaración de los nodos de la aplicación
server web1 web1:80 check
```

Recarga en caliente mediante el archivo de configuración

Abre tu interfaz de estadísticas (puerto 8404). Normalmente deberías ver 1 servidor activo en el backend **default**. El objetivo es añadir el servidor **web2** a este pool en caliente, es decir, sin reiniciar el contenedor de HAProxy, para conservar las conexiones en curso. Los pasos:

- Ver la fecha de creación de los contenedores de HAProxy: la información está en la columna 'CREATED'

```
sudo docker compose ps haproxy
```

- Edita el archivo para añadir el segundo servidor y recarga el servicio:

```
sudo docker compose kill -s HUP haproxy
```

- Verifica en la interfaz que el nuevo servidor aparece y que la fecha de creación del contenedor no ha cambiado

Corrección :

1. Añade el servidor **web2** en el bloque backend default del archivo de configuración cargado por HAProxy:

```
server web2 web2:80 check
```

2. Para recargar la configuración sin reiniciar el contenedor, envía una señal de recarga (SIGHUP) al proceso de HAProxy:

```
sudo docker compose kill -s HUP haproxy
```

Verifica en el dashboard de estadísticas: el backend **default** debe mostrar ahora 2 servidores sin que el Uptime del servicio se haya reiniciado.

Recarga en caliente mediante socat

HAProxy dispone de una **Runtime API** accesible a través de un socket Unix. Permite modificar el estado del balanceador (pesos, estado de los servidores, adición dinámica) sin ninguna recarga.

Para inspeccionar el estado interno o lanzar comandos directamente en tu contenedor, abre un shell con:
`sudo docker exec -it <nom_du_conteneur> sh`

Para usar la API, debes haber activado el socket en la sección **global** de la configuración cargada por HAProxy:
`stats socket /tmp/admin.sock mode 600 level admin`

Lectura recomendada: Para profundizar en la gestión dinámica de backends y servidores mediante la Runtime API, consulta nuestro artículo dedicado: [HAProxy: añadir y eliminar backends de forma dinámica](https://elearning.rousselTM.fr/HAProxy-a%C3%A9nadir-y-eliminar-backends-de-forma-din%C3%A1mica). Al añadirlo, deberías ver aparecer el servidor en tu página de estadísticas. Pero, como habrás comprobado, socat es un comando local, así que hay que ejecutarlo en todas las instancias de tu clúster HAProxy. Además, como indica la documentación, añadir un servidor no basta para que reciba tráfico: hay que activarlo...

Corrección :

1. Conéctate al contenedor de HAProxy:

```
docker exec -it <nom_du_conteneur> sh
```

2. Usa **socat** para enviar el comando de adición al motor:

```
echo "add server default/web3 web3:80 check" | socat stdio /tmp/admin.sock
```

3. Por defecto, un servidor añadido dinámicamente está en modo mantenimiento. Actívalo:

```
echo "enable server default/web3" | socat stdio /tmp/admin.sock
```

4. También hay que activar el healthcheck:

```
echo "enable health default/web3" | socat stdio /tmp/admin.sock
```

Verifica en el dashboard: el servidor web3 aparece al instante. Pero cuidado con socat, que no gestiona la persistencia y se ejecuta localmente

El Backend web_servers

Crea un grupo de servidores llamado 'web_servers'. Este backend debe funcionar en modo HTTP y usar el algoritmo de reparto 'least connection', con un peso para cada servidor que corresponda a su id de servidor. También debes añadir una regla para transmitir al backend la dirección IP real del cliente mediante el encabezado 'X-Real-IP' y declarar los 3 nodos de la aplicación (web1, web2, web3) en el puerto 80 con comprobación de salud (check). Por último, actualiza el frontend http_in para que envíe el tráfico por defecto a este backend.

Corrección :

```
frontend http_in
    bind :80
    default_backend web_servers

backend web_servers
    mode http
    balance leastconn
    http-request set-header X-Real-IP %[src]
    # Declaración de los 3 nodos de la aplicación del clúster
    server web1 web1:80 weight 1 check
    server web2 web2:80 weight 2 check
    server web3 web3:80 weight 3 check
```

Escalado de HAProxy

El escalado horizontal (scaling) consiste en multiplicar las instancias de HAProxy para garantizar la **alta disponibilidad (HA)**. En una arquitectura real, tener una sola instancia de balanceador crea un 'Single Point of Failure' (SPOF): si HAProxy cae, toda la aplicación queda inaccesible, aunque los servidores web estén sanos. En un entorno Docker, el escalado permite repartir la carga de red y asegurar la continuidad del servicio durante las actualizaciones (rolling updates). Lanza el comando de Docker Compose con el parámetro `--scale haproxy=3` para que el número de instancias de HAProxy pase a 3.

Nota: Multiplicar el número de nodos de HAProxy no basta para que esté en alta disponibilidad. Necesita una VIP: en nuestro caso, es Nginx quien hará el balanceo de carga para asegurar una alta disponibilidad completa.

Corrección :

Para escalar tu punto de entrada en el mundo Docker sin modificar tu archivo de configuración `compose.yml`, usamos el parámetro de ejecución `--scale`:

Despliegue: Aplica la nueva topología:

```
sudo -E docker compose up -d --scale haproxy=3
```

¿Por qué es importante?

- **Resiliencia:** Si un contenedor de HAProxy falla, las demás instancias siguen respondiendo.
- **Mantenimiento en caliente:** Puedes reiniciar o actualizar una instancia sin interrumpir el tráfico global.
- **Rendimiento:** En producción, estas instancias estarían repartidas en distintos nodos físicos (con Docker Swarm o Kubernetes), lo que permitiría absorber caudales de red masivos.

Nota: En local con Docker Compose, las instancias pueden compartir el mismo mapeo de puertos si usas una red de tipo 'overlay' o un balanceador de carga previo.

Sincronización de sesiones con el 'Peering'

En un clúster HAProxy de alta disponibilidad, una conmutación entre dos nodos puede provocar la pérdida de la información de persistencia si cada instancia conserva sus datos localmente. Para evitar este problema, HAProxy integra un mecanismo nativo llamado **Peers** que replica en tiempo real el contenido de las *Stick Tables* entre varios servidores HAProxy. Esta funcionalidad garantiza que la información de sesión, los contadores de limitación de tasa (Rate Limiting), las detecciones de abusos o los datos de afinidad de cliente sigan siendo coherentes tras un failover.

IMPORTANTE: El peering no sincroniza las sesiones de aplicación (PHP, Java, .NET, etc.). Solo replica los datos almacenados en las *Stick Tables* de HAProxy. Para compartir las sesiones de aplicación, se recomienda usar una solución externa como Redis o Memcached.

¿Qué se puede sincronizar con los Peers?

- **Persistencia de sesión:** mantener la afinidad entre un cliente y su servidor backend.
- **Rate Limiting:** compartir los contadores de peticiones entre los nodos de HAProxy.
- **Protección contra abusos:** detección distribuida de ataques o comportamientos sospechosos.
- **Seguimiento de usuarios:** conservar información relacionada con los clientes incluso tras una conmutación.

Los Peers son especialmente útiles en arquitecturas que usan Keepalived o VRRP, donde una dirección IP virtual puede migrar de un HAProxy a otro.

Buenas prácticas:

- Usa una red dedicada o aislada para los intercambios de peering, para evitar interferencias con el tráfico de los usuarios.

- Verifica que los puertos usados por los Peers son accesibles entre los nodos del clúster.
- Mantén una definición idéntica de la sección peers en todos los servidores HAProxy.
- Supervisa el estado de sincronización mediante la Runtime API o la página de estadísticas de HAProxy.
- Para las aplicaciones con estado, aun así es preferible externalizar las sesiones en Redis para evitar cualquier dependencia del balanceador.

Debes añadir una sección para compartir las sesiones llamada 'haproxy_cluster' que use los tres servidores HAProxy haproxy-formation-haproxy-1/2/3 en el puerto por defecto. Después, debes activar este uso compartido de sesiones en el backend 'web_servers' con un tamaño de 100k, una expiración a los 30m y basándote en la IP de origen del visitante.

Corrección :

Cada miembro del clúster debe declarar la misma sección peers con la lista completa de los nodos que participan en la replicación. Después, las Stick Tables deben asociarse a este grupo de Peers para que su contenido se sincronice automáticamente.

```
peers haproxy_cluster
    peer haproxy1 haproxy-formation-haproxy-1:1024
    peer haproxy2 haproxy-formation-haproxy-2:1024
    peer haproxy3 haproxy-formation-haproxy-3:1024

backend web_servers
    # Tabla sincronizada entre todos los HAProxy
    stick-table type ip size 100k expire 30m peers haproxy_cluster

    # Afinidad basada en la dirección IP de origen
    stick on src

    # La configuración anterior
    mode http
    balance leastconn
    http-request set-header X-Real-IP %[src]
    server web1 web1:80 weight 1 check
    server web2 web2:80 weight 2 check
    server web3 web3:80 weight 3 check
```

Enrutamiento avanzado con ACL

Redirigir las peticiones que empiezan por /api al servidor 2.

Modificación de las ACL

El enrutamiento inteligente permite dirigir el tráfico a distintos backends según criterios precisos. En este ejercicio, debes reconfigurar HAProxy para que identifique las peticiones destinadas a la API (que empiezan por la ruta /api) y las envíe exclusivamente al backend secure_servers, que contiene los servidores del servicio **web**, pero solo de la 2.^a a la 10.^a instancia. El resto del tráfico debe seguir repartiéndose mediante el backend por defecto.

Por defecto, HAProxy hace pruebas de nivel 4 (TCP - IP:PUERTO), pero debes modificar el backend para hacer una comprobación de salud de nivel 7 (HTTP - página web) que verifique que la página 'secure.html' devuelve la cadena 'Service\ '

Corrección :

Aquí tienes un ejemplo de configuración (se aconseja repartirlo en varios archivos) con un bloque para el frontend, otro para la resolución de los servicios de Docker y otro para el backend

```
frontend http_in
    bind *:80
    # Definimos una ACL que comprueba si la ruta empieza por /api
    acl is_api path_beg /api
    # Pedimos a HAProxy que use un backend específico si la ACL es verdadera
    use_backend secure_servers if is_api
    default_backend web_servers

resolvers docker
    # DNS interno de Docker (permite resolver los nombres de los contenedores)
    nameserver dns 127.0.0.11:53
    # Numero de intentos en caso de fallo DNS
    resolve_retries 3
    # Timeout global de una petición DNS
    timeout resolve 1s
    # Tiempo de espera entre dos intentos de resolución
    timeout retry 1s
    # Tiempo durante el que una resolución se considera válida
    # Importante para evitar re-resoluciones demasiado frecuentes
    hold valid 10s

backend secure_servers
    # Healthcheck HTTP personalizado
    option httpchk GET /secure.html
    # Verifica que la respuesta contiene 'Service'
    http-check expect string Service\
    # Pool dinámico seguro
    server-template web 2-10 web:80 check resolvers docker init-addr libc,none
```

Lista blanca de IP (whitelisting)

El filtrado por dirección IP es un pilar de la seguridad. Ahora debes restringir el acceso al dashboard de estadísticas (puerto 8404) para autorizar solo a las fuentes internas que lo necesitan: en nuestro caso, el servidor Prometheus (DNS prometheus en Docker) y los administradores.

Whitelisting vs Blacklisting:

HAProxy es compatible con ambos enfoques mediante sus reglas de ACL.

- **Whitelisting (lista blanca):** estrategia *'Default Deny'*. Se bloquea todo por defecto y solo se autorizan las fuentes conocidas. Es el enfoque más robusto para la seguridad.
Casos de uso: acceso a una consola de administración, API interna reservada a socios, restricción del acceso a las IP de una VPN corporativa.
- **Blacklisting (lista negra):** estrategia *'Default Allow'*. Se deja pasar todo el tráfico salvo las IP identificadas como maliciosas.
Casos de uso: bloqueo de bots conocidos (scrapers), protección temporal contra una IP que realiza un ataque de fuerza bruta, o geobloqueo (bloqueo de regiones enteras).

Corrección :

En un entorno real, no olvides añadir las IP de tus pasarelas VPN o de tus oficinas. En nuestra infraestructura de TP, los servicios internos de Docker usan la subred del proyecto; por eso la autorizamos, para conservar el acceso desde Prometheus y desde el Nginx de administración. Así se aplica este filtrado en el frontend de estadísticas:

```
frontend admin
  bind :8404
  # Definición de la ACL basada en la IP de origen (src)
  acl allowed_monitors src nginx-admin prometheus
  # Se deniega el acceso si la IP no está autorizada
  http-request deny if !network_allowed

  stats enable
  stats uri /
  stats refresh 5s
```

Truco de factorización: Si tienes una lista de IP que quieres reutilizar en varios frontends, evita copiarlas y pegarlas. Usa un archivo externo (p. ej.: /etc/haproxy/whitelist.lst) que contenga una IP o un CIDR por línea. Tu ACL se vuelve entonces sencilla y centralizada:

```
acl network_allowed src -f /etc/haproxy/whitelist.lst
```

Seguridad y endurecimiento

Protección del clúster contra los abusos y los ataques de red.

Protección contra DoS y DDoS

Configura el frontend http_in para limitar cada IP de origen a un máximo de 15 conexiones TCP simultáneas (capa 4). Una vez aplicada la configuración, intenta abrir varias conexiones en paralelo para verificar que HAProxy rechaza las siguientes.

Verificación: Simula carga con un bucle de shell que abra numerosas conexiones en segundo plano:

```
for i in $(seq 1 20); do curl -s http://localhost & done
```

Mira tus logs de HAProxy o el dashboard: las conexiones por encima de 15 deben rechazarse inmediatamente.

DoS vs DDoS: ¿cuál es la diferencia?

- **DoS (Denial of Service):** el ataque procede de una sola máquina. HAProxy puede bloquearlo fácilmente limitando las conexiones por IP.
- **DDoS (Distributed DoS):** el ataque procede de miles de fuentes (botnet). Bloquear una IP ya no basta, porque el tráfico está distribuido.

Recomendaciones de HAProxy ([Leer el artículo oficial](#)):

- **Defensa en profundidad:** para los ataques volumétricos masivos (capas 3 y 4), usa servicios de protección en la nube (Cloudflare, AWS Shield) por delante.
- **Filtrado en capa 7:** usa HAProxy para identificar firmas de ataque específicas en los encabezados o las URL mediante ACL.
- **Stick Tables:** siguen siendo esenciales para limitar el impacto de cada nodo de la botnet en los recursos de tu aplicación.

Corrección :**Procedimiento:**

1. Modifica el frontend `http_in` existente para integrar el seguimiento de las conexiones:

```
frontend http_in
  bind :80
  # Creación de la tabla de seguimiento
  stick-table type ip size 100k expire 30s store conn_cur
  # Seguimiento de la IP de origen
  tcp-request connection track-sc0 src
  # Rechazo por encima de 15 conexiones
  tcp-request connection reject if { sc0_conn_cur gt 15 }
  default_backend web_servers
```

2. Recarga HAProxy.

Protección contra Slowloris

El ataque Slowloris satura el servidor enviando encabezados HTTP muy lentamente. Implementa una protección limitando el tiempo de recepción de los encabezados a 5 segundos.

Simulación de ataque: Lanza este comando **antes** de la configuración (la conexión se quedará bloqueada) y luego **después** (HAProxy cortará la conexión al cabo de 5 s):

```
curl -X GET "http://localhost:80/" \
  -H "X-Slow: $(head -c 5000 /dev/urandom | base64)" --limit-rate 1 \
  --verbose
```

Explicación de los parámetros:

- `-H "X-Slow: ..."`: genera un encabezado enorme de 5000 bytes (con head y tr) para forzar un envío largo.
- `--limit-rate 1`: limita el envío a 1 byte/s. Sin protección, la conexión permanecería abierta más de una hora.
- `--verbose`: permite observar en directo el momento exacto en que HAProxy interrumpe la sesión.

Recomendaciones de HAProxy ([Leer el artículo oficial](#)): Para una protección óptima, HAProxy recomienda combinar varios ajustes:

- `timeout http-request 5s`: limita el tiempo de espera de los encabezados (defensa principal).
- `timeout http-keep-alive 1s`: libera rápidamente los slots de conexión inactivos.
- `maxconn`: limita el número global de conexiones para proteger los descriptores de archivo del sistema contra el agotamiento.

Corrección :**Procedimiento:**

1. En la sección `defaults` (o en el frontend), añade la directiva de `timeout`:

```
defaults
# Corta la conexión si los encabezados tardan más de 5 s en llegar
timeout http-request 5s
#
timeout http-keep-alive 1s
#
maxconn 10000
```

2. Recarga HAProxy y vuelve a lanzar el comando de simulación. Deberías ver un mensaje Empty reply from server o un cierre del socket al cabo de unos 5 segundos.

Limitar las peticiones (Rate Limiting)

Implanta una cuota de 50 peticiones como máximo cada 10 segundos por IP. Prueba a superar el umbral y verifica que HAProxy devuelve un código de error HTTP 429.

Verificación: Ejecuta una ráfaga de peticiones curl para agotar la cuota:

```
for i in $(seq 1 60); do curl -I http://localhost; done
```

Deberías observar respuestas HTTP/1.1 429 Too Many Requests a partir de la petición 50.

Corrección :

Procedimiento:

1. Actualiza la stick-table del frontend http_in para seguir a la vez las conexiones en curso y la tasa de peticiones (http_req_rate):

```
frontend http_in
bind :80
# Se conserva conn_cur y se añade el almacenamiento de la tasa en 10s
stick-table type ip size 100k expire 30s store conn_cur,http_req_rate(10s)
tcp-request connection track-sc0 src
tcp-request connection reject if { sc0_conn_cur gt 15 }
# Seguimiento a nivel HTTP (sc1)
http-request track-sc1 src
# Bloqueo con código 429 (Too Many Requests)
http-request deny deny_status 429 if { sc1_http_req_rate gt 50 }
default_backend web_servers
```

2. Recarga la configuración.

Módulos anti-bot y detección de scraping

Bloquea los recolectores de datos (scrapers) y los bots maliciosos identificando sus firmas de User-Agent sospechosas.

Simulación de bot: Intenta acceder al sitio simulando un bot conocido con curl:

```
curl -I -H "User-Agent: Googlebot-Scraper" http://localhost/
```

Corrección :

Procedimiento:

1. Añade estas ACL y esta regla en el frontend http_in existente, conservando las protecciones ya configuradas:

```
# Detección de palabras clave en el User-Agent (sin distinguir mayúsculas)
acl is_bot hdr_sub(user-agent) -i bot crawler scraper spider
# Rechazo de las peticiones identificadas como bots
http-request deny deny_status 403 if is_bot
```

2. Recarga HAProxy y verifica que tu petición curl con el encabezado sospechoso se rechaza ahora con un código 403.

Terminación silenciosa de peticiones

El 'Silent Drop' es una técnica de defensa avanzada: en lugar de enviar un código de error (403/429) que confirma al atacante la existencia de una protección, HAProxy simplemente ignora la petición. El atacante malgasta sus recursos esperando una respuesta que nunca llegará.

Verificación: Prueba a acceder a una ruta prohibida y comprueba que curl se queda esperando (hang) hasta el timeout:

```
curl -v http://localhost/admin/config
```

Corrección :**Procedimiento:**

1. Añade la acción silent-drop en el frontend http_in existente para las rutas o flujos prohibidos:

```
acl is_forbidden path_beg /admin
# Cierre de la conexión sin enviar ni un solo bit de respuesta
http-request silent-drop if is_forbidden
```

2. Recarga la configuración. El cliente no recibirá ningún dato (ni siquiera un TCP RST), lo que resulta especialmente eficaz contra los escáneres automáticos.

Observabilidad

Puesta en marcha de la monitorización, la exposición de métricas y la telemetría.

Stats Dashboard

Verifica que el punto de entrada de administración configurado antes sigue exponiendo el dashboard de estadísticas en la raíz del puerto 8404.

Corrección :

```
frontend admin
  bind :8404
  stats enable
  stats uri /
```

```
stats refresh 5s
```

Prometheus: exposición de las métricas

Configura HAProxy para exponer las métricas de rendimiento en formato Prometheus en la URI /metrics. Así un servidor Prometheus puede hacer scraping del estado del balanceador.

Verificación: Prueba a recuperar las métricas con curl:

```
curl http://localhost:8404/metrics
```

Deberías ver desfilas las métricas en formato de texto plano (HELP y TYPE).

Corrección :

Procedimiento:

1. En tu bloque frontend admin, usa la directiva http-request use-service:

```
frontend admin
  bind :8404
  stats enable
  stats uri /
  stats refresh 5s
  # Exposición de las métricas para Prometheus
  # Esta regla intercepta las peticiones a la URI '/metrics' y devuelve el formato Prometheus
  http-request use-service prometheus-exporter if { path /metrics }
```

2. Recarga la configuración. HAProxy sirve ahora las métricas en el mismo puerto que el dashboard.

Compatibilidad con OpenTelemetry (OTel)

Configura la exportación nativa de datos de telemetría hacia un colector de OpenTelemetry.

Nota: La compatibilidad nativa con OpenTelemetry está disponible a partir de la versión **3.4** de HAProxy.

Corrección :

Procedimiento:

1. Declara el exportador OTel en la sección global:

```
global
  otel-exporter my-collector
  endpoint otel-collector:4317
```

2. Activa el envío de trazas en tu frontend para propagar el contexto:

```
frontend http_in
  http-request otel-send-trace
```

Práctica: persistencia, health checks y página de estadísticas (Trabajos Prácticos)

Configurar un backend con cookies de sesión, sondas de salud y estadísticas.

Backend resiliente

Contexto. Un sitio de comercio electrónico pierde carritos cuando el usuario cambia de servidor y muestra errores cuando un servidor cae. Una cookie de persistencia y los health checks resuelven ambos problemas; la página de estadísticas muestra lo que ocurre en tiempo real.

Requisitos: Docker · **Duración estimada:** 40 min

Tres servidores web detrás de HAProxy.

1. Entorno

¿Por qué? Una red Docker compartida permite a HAProxy alcanzar los servidores por nombre, sin gestionar direcciones IP.

Inicia HAProxy y tres contenedores traefik/whoami en la misma red.

Corrección :

```
docker network create lab
for i in 1 2 3; do docker run -d --name web$i --network lab traefik/whoami; done
```

2. Configuración con cookie y health check

¿Por qué? La cookie garantiza que un usuario permanece en el mismo servidor y la sonda retira automáticamente un servidor defectuoso de la rotación.

Persistencia por cookie, sonda HTTP y página de estadísticas en 8404.

Corrección :

```
# haproxy.cfg
global
  stats socket /tmp/haproxy.sock mode 660 level admin
defaults
  mode http
  timeout connect 5s
  timeout client 30s
  timeout server 30s
frontend fe
  bind *:80
  default_backend app
backend app
  balance roundrobin
  cookie SRV insert indirect nocache
  option httpchk GET /health
```

```

default-server inter 2s fall 2 rise 2
server web1 web1:80 check cookie s1
server web2 web2:80 check cookie s2
server web3 web3:80 check cookie s3
listen stats
bind *:8404
stats enable
stats uri /stats
stats refresh 5s

docker run -d --name lb --network lab -p 8080:80 -p 8404:8404 -v $PWD/haproxy.cfg:/usr/local/etc/haproxy/haproxy.cfg:ro haproxy:3.0
docker logs lb

```

3. Probar la persistencia

¿Por qué? Comparar con y sin cookie muestra la diferencia entre un balanceo rotativo y uno «pegajoso».

Sin cookie la carga rota; con la cookie, siempre el mismo servidor.

Corrección :

```

for i in 1 2 3 4; do curl -s localhost:8080 | grep Hostname; done
curl -s -c jar localhost:8080 >/dev/null; for i in 1 2 3 4; do curl -s -b jar localhost:8080 |
grep Hostname; done

```

4. Simular una caída

¿Por qué? Simular una caída es la única forma de probar que la alta disponibilidad funciona; la página de estadísticas y la Runtime API muestran el estado en directo.

Detén web2 y observa la página de estadísticas y la Runtime API.

Corrección :

```

docker stop web2
sleep 5; curl -s 'localhost:8404/stats;csv' | cut -d, -f1,2,18 | grep -E 'app,web'
docker exec lb sh -c 'echo "show servers state app" | socat stdio /tmp/haproxy.sock' | awk '{p
rint $4, $6}'
docker start web2

```

Práctica: terminación SSL y enrutamiento por ACL (Trabajos Prácticos)

Servir HTTPS, redirigir HTTP y enrutar según la ruta y el host.

HTTPS y ACL

Contexto. Un sitio debe pasar a HTTPS, redirigir el antiguo HTTP y separar el tráfico de la API del resto, restringiendo el acceso a una interfaz de administración. HAProxy termina el TLS y aplica ACL para enrutar y filtrar.

Requisitos: Práctica anterior, OpenSSL · **Duración estimada:** 45 min

Un certificado autofirmado y reglas de enrutamiento.

1. Generar un certificado

¿Por qué? HAProxy espera un archivo PEM con certificado y clave: generarlo uno mismo hace la práctica autónoma, sin autoridad externa.

Crea el archivo PEM que espera HAProxy.

Corrección :

```
mkdir -p certs
openssl req -x509 -newkey rsa:2048 -nodes -days 30 -keyout certs/k.key -out certs/c.crt -subj
'/CN=app.localhost'
cat certs/c.crt certs/k.key > certs/app.pem
```

2. Frontend HTTPS y redirección

¿Por qué? La terminación TLS descarga el cifrado de los servidores y centraliza certificados; la redirección y HSTS fuerzan un uso seguro.

Escucha en 443, redirige 80 a HTTPS y añade HSTS.

Corrección :

```
frontend fe
  bind *:80
  bind *:443 ssl crt /usr/local/etc/haproxy/certs/app.pem alpn h2,http/1.1
  http-request redirect scheme https unless { ssl_fc }
  http-response set-header Strict-Transport-Security "max-age=31536000"
  acl is_api path_beg /api
  acl is_admin hdr(host) -i admin.localhost
  acl lan src 127.0.0.1 172.16.0.0/12
  http-request deny if is_admin !lan
  use_backend api if is_api
  default_backend web
backend api
  server api1 api:80 check
```

```
backend web
  server web1 web:80 check
```

3. Probar el enrutamiento

¿Por qué? Probar cada ruta y el certificado verifica que las ACL enrutan bien y que el cliente ve lo esperado.

Llama a / y /api y comprueba el certificado.

Corrección :

```
curl -sk https://localhost/ | head -3
curl -sk https://localhost/api | head -3
curl -sI http://localhost | grep -i location
openssl s_client -connect localhost:443 -servername app.localhost </dev/null 2>/dev/null | ope
nssl x509 -noout -subject -dates
```

4. Validar y recargar sin cortes

¿Por qué? Validar antes de recargar y usar una señal de recarga evita interrumpir las conexiones en curso.

Modifica una regla y recarga limpiamente.

Corrección :

```
docker exec lb haproxy -c -f /usr/local/etc/haproxy/haproxy.cfg
docker kill -s HUP lb      # rechargement sans coupure
```

[HAProxy Ingress Controller para Kubernetes](#) (Trabajos Prácticos)

HAProxy Ingress Controller para Kubernetes

[Ingress Controller: guía de producción](#) (Trabajos Prácticos)

Ingress Controller: guía de producción

[Buenas prácticas de HAProxy](#) (Trabajos Prácticos)

Buenas prácticas de HAProxy

[Novedades de HAProxy: OpenTelemetry, JWT, rendimiento](#) (Trabajos Prácticos)

Novedades de HAProxy: OpenTelemetry, JWT, rendimiento

Cuestionario (QCM)

Información importante

Para validar tus conocimientos y consultar las correcciones interactivas de este cuestionario, te invitamos a conectarte a nuestra plataforma en línea: elearning.rousselTM.fr.

Q1 : ¿Qué significa 'HA' en HAProxy?

1. High Accuracy
2. High Availability
3. Hyper Access
4. Home Automation

? **Pista** : HAProxy significa High Availability Proxy.

Artículos para profundizar :

Q2 : ¿En qué capa del modelo OSI puede operar HAProxy?

1. Capas 2 y 3
2. Solo capa 4
3. Solo capa 7
4. Capa 4 y capa 7

? **Pista** : HAProxy es un balanceador capaz de tratar el tráfico a nivel TCP (capa 4) y HTTP (capa 7).

Artículos para profundizar :

Q3 : ¿Qué sección del archivo de configuración define los parámetros de seguridad del proceso (user, group)?

1. defaults
2. frontend
3. global
4. backend

? **Pista** : La sección 'global' gestiona los parámetros del propio proceso del sistema.

Artículos para profundizar :

Q4 : ¿Qué comando permite comprobar la sintaxis del archivo haproxy.cfg?

1. haproxy -check
2. haproxy -c -f /etc/haproxy/haproxy.cfg
3. haproxy -t
4. systemctl test haproxy

? Pista : La opción -c (check) combinada con -f (file) permite validar la configuración.

Artículos para profundizar :

Q5 : ¿En qué bloque se define la dirección IP y el puerto de escucha?

1. backend
2. global
3. defaults
4. frontend

? Pista : El 'frontend' es la puerta de entrada donde se usa la directiva 'bind'.

Artículos para profundizar :

Q6 : ¿Qué algoritmo de reparto distribuye las peticiones por turnos?

1. leastconn
2. source
3. roundrobin
4. uri

? Pista : Round Robin fue el algoritmo cíclico por defecto hasta la versión 3.3, que pasó a 'random'.

Artículos para profundizar :

Q7 : ¿Qué algoritmo se recomienda para sesiones largas como SQL o WebSocket?

1. roundrobin
2. leastconn
3. static-rr
4. first

? Pista : *Leastconn envía el tráfico al servidor con menos conexiones activas, ideal para sesiones persistentes.*

Artículos para profundizar :

Q8 : ¿Qué directiva activa los health checks en un servidor?

1. option health
2. check
3. verify
4. test-alive

? Pista : *Se añade la palabra clave 'check' al final de la línea 'server'.*

Artículos para profundizar :

Q9 : ¿Qué hace la directiva 'option httpchk'?

1. Activa el modo HTTP
2. Realiza una petición HTTP para comprobar la salud del backend
3. Registra las peticiones HTTP
4. Redirige HTTP a HTTPS

? Pista : *Permite pasar de un simple check TCP a una verificación HTTP de aplicación.*

Artículos para profundizar :

Q10 : ¿Cómo se llama descifrar el SSL en HAProxy para enviar HTTP en claro a los servidores?

1. SSL Passthrough
2. SSL Bridging
3. SSL Termination (Offloading)
4. SSL Encrypt

? Pista : La terminación SSL descarga a los servidores web del cálculo del cifrado.

Artículos para profundizar :

Q11 : ¿Qué palabra clave se usa para crear una regla de enrutamiento basada en la URL?

1. RULE
2. IF
3. ACL
4. MATCH

? Pista : ACL (Access Control List) es el motor de decisión de HAProxy.

Artículos para profundizar :

Q12 : ¿Qué ACL comprueba si la ruta empieza por '/api'?

1. path_end /api
2. url_beg /api
3. path_beg /api
4. path_is /api

? Pista : path_beg comprueba el inicio de la ruta (path).

Artículos para profundizar :

Q13 : ¿Qué servicio de HAProxy gestiona de forma centralizada varias instancias Enterprise?

1. HAProxy OSS
2. HAProxy Fusion
3. HAProxy Edge
4. HAProxy One

? Pista : *HAProxy Fusion es el plano de control para la gestión multiclúster.*

Artículos para profundizar :

Q14 : En HAProxy One, ¿qué es HAProxy Edge?

1. Un software de monitorización
2. Una red de distribución global (ADN/CDN)
3. Una versión para Raspberry Pi
4. Un plugin para Kubernetes

? Pista : *HAProxy Edge es la capa ADN distribuida a escala mundial.*

Artículos para profundizar :

Q15 : ¿Qué directiva de 'global' permite usar varios núcleos de CPU?

1. nbproc
2. nbthread
3. cpu-map
4. multi-core

? Pista : *nbthread es el método moderno recomendado para el multihilo.*

Artículos para profundizar :

Q16 : ¿Cómo forzar la redirección de HTTP a HTTPS?

1. redirect scheme https if ![ssl_fc]
2. force https
3. option https-only
4. http-request redirect https

? Pista : *redirect scheme https es la sintaxis clásica para redirigir si la conexión no es segura.*

Artículos para profundizar :

Q17 : ¿Qué significa el estado 'L7OK' en las estadísticas?

1. El servidor responde en TCP
2. El check HTTP ha funcionado
3. El servidor está en mantenimiento
4. La capa 7 está desactivada

? Pista : *L7OK indica que el check de capa 7 (aplicación) es válido.*

Artículos para profundizar :

Q18 : ¿Qué puerto se usa habitualmente para la interfaz de estadísticas de HAProxy?

1. 80
2. 443
3. 8404
4. 8080

? Pista : *Aunque es configurable, el puerto 8404 es el estándar recomendado.*

Artículos para profundizar :

Q19 : ¿Qué directiva vincula a un usuario con un servidor concreto durante su sesión?

1. persistence
2. stick-table
3. cookie insert
4. session-bind

? Pista : *'cookie insert' gestiona la persistencia mediante una cookie inyectada por HAProxy.*

Artículos para profundizar :

Q20 : ¿Cuál es la principal diferencia entre HAProxy OSS y Enterprise?

1. El rendimiento
2. El soporte comercial y los módulos de seguridad (WAF)
3. El lenguaje de programación
4. La compatibilidad con Linux

? Pista : *La versión Enterprise aporta soporte, estabilidad garantizada y módulos como un WAF avanzado.*

Artículos para profundizar :

Q21 : ¿Qué hace el modo 'mode tcp'?

1. Analiza las cabeceras HTTP
2. Hace balanceo ciego a nivel de transporte
3. Activa el cortafuegos
4. Acelera la descarga

? Pista : *El modo TCP trata los flujos sin mirar el contenido de aplicación (capa 4).*

Artículos para profundizar :

Q22 : ¿Qué palabra clave define un servidor de reserva que solo se usa si los demás fallan?

1. secondary
2. backup
3. failover
4. spare

? Pista : La palabra clave 'backup' activa el servidor solo en caso de caída del grupo principal.

Artículos para profundizar :

Q23 : ¿Cómo limitar el número máximo de conexiones globales?

1. max-limit
2. conn-limit
3. maxconn
4. limit-sessions

? Pista : La directiva 'maxconn' se usa tanto en global como en frontend/backend.

Artículos para profundizar :

Q24 : ¿Para qué sirve la 'Runtime API'?

1. Para modificar la configuración sin recargar el servicio
2. Para crear aplicaciones web
3. Para actualizar HAProxy
4. Para generar certificados

? Pista : La Runtime API permite cambiar pesos, vaciar servidores o leer estadísticas de forma dinámica.

Artículos para profundizar :

Q25 : ¿Qué sección transmite sus parámetros a los frontends y backends?

1. global
2. main
3. defaults
4. common

? Pista : La sección 'defaults' evita repetir los timeouts y las opciones de logs.

Artículos para profundizar :

Q26 : ¿Para qué es útil el algoritmo 'source'?

1. Repartir de forma equitativa
2. Garantizar que una IP de cliente vaya siempre al mismo servidor
3. El streaming de vídeo
4. Solo el tráfico móvil

? Pista : Crea una persistencia basada en el hash de la dirección IP de origen.

Artículos para profundizar :

Q27 : ¿Qué directiva añade la cabecera 'X-Forwarded-For'?

1. add-header xff
2. option forwardfor
3. http-request set-xff
4. forward-ip

? Pista : Es la opción estándar para transmitir la IP real del cliente al servidor backend.

Artículos para profundizar :

Q28 : ¿Qué significa 'inter 2s' en una línea server?

1. Timeout de 2 segundos
2. Intervalo de 2 segundos entre dos health checks
3. Espera de 2 segundos antes de reconectar
4. Duración de la sesión

? Pista : *Es el intervalo regular entre cada prueba de salud.*

Artículos para profundizar :

Q29 : ¿Qué directiva define el backend por defecto de un frontend?

1. use_backend
2. default_backend
3. backend_link
4. main_backend

? Pista : *Es el destino que se usa si ninguna regla ACL coincide.*

Artículos para profundizar :

Q30 : ¿Se pueden usar varios archivos de configuración con HAProxy?

1. No, solo es posible un archivo haproxy.cfg
2. Sí, apuntando -f a un directorio
3. Sí, solo en la versión Enterprise
4. Solo mediante 'include'

? Pista : *HAProxy puede concatenar varios archivos pasados con -f, o un directorio.*

Artículos para profundizar :

Q31 : ¿Qué hace 'stats refresh 5s'?

1. Reinicia HAProxy cada 5 s
2. Actualiza automáticamente la página de estadísticas cada 5 s
3. Comprueba los servidores cada 5 s
4. Limpia los logs

? Pista : Es una opción de comodidad para la monitorización en tiempo real desde el dashboard.

Artículos para profundizar :

Q32 : ¿Qué comando recarga HAProxy sin interrupción?

1. systemctl restart haproxy
2. systemctl reload haproxy
3. haproxy -update
4. service haproxy stop && start

? Pista : El 'reload' hace una reconfiguración en caliente transfiriendo los sockets a los nuevos procesos.

Artículos para profundizar :

Q33 : ¿Para qué sirve 'fall 3' en un check?

1. El servidor cae al cabo de 3 segundos
2. El servidor se marca DOWN tras 3 fallos consecutivos
3. Hacen falta 3 servidores para que funcione
4. El peso del servidor se divide entre 3

? Pista : Es el umbral de tolerancia antes de considerar el servidor como defectuoso.

Artículos para profundizar :

Q34 : ¿Y 'rise 2'?

1. Aumenta el peso
2. El servidor vuelve a UP tras 2 éxitos consecutivos
3. Retardo de arranque
4. Número de conexiones

? Pista : *Es el número de pruebas correctas necesarias para reintegrar un servidor en el grupo.*

Artículos para profundizar :

Q35 : ¿Para qué sirve la opción 'http-server-close'?

1. Cerrar el servidor
2. Desactivar el Keep-Alive del lado del servidor
3. Bloquear las peticiones
4. Optimizar el SSL

? Pista : *Permite liberar las conexiones de los servidores más rápido.*

Artículos para profundizar :

Q36 : ¿Qué directiva bloquea una dirección IP?

1. block ip
2. http-request deny if { src 1.2.3.4 }
3. acl ban ip
4. deny_access 1.2.3.4

? Pista : *http-request deny es la herramienta estándar para filtrar peticiones.*

Artículos para profundizar :

Q37 : ¿Cómo gestiona HAProxy Enterprise las actualizaciones de seguridad?

1. No lo hace
2. Mediante correcciones retroportadas a versiones LTS
3. Solo cambiando de versión mayor
4. Automáticamente sin avisar

? Pista : *El soporte Enterprise garantiza correcciones en versiones estables a largo plazo.*

Artículos para profundizar :

Q38 : ¿Qué es una 'Stick Table'?

1. Una mesa de madera
2. Una base de datos en memoria para seguir contadores (IP, sesiones, errores)
3. Un archivo de log
4. Un algoritmo de hash

? Pista : *Es una estructura de datos ultrarrápida en RAM para el seguimiento avanzado.*

Artículos para profundizar :

Q39 : ¿Se puede balancear una base de datos MySQL con HAProxy?

1. No, solo HTTP
2. Sí, en modo TCP (capa 4)
3. Solo con un plugin especial
4. Sí, pero sin health check

? Pista : *HAProxy destaca en el proxy de protocolos binarios como MySQL o PostgreSQL.*

Artículos para profundizar :

Q40 : ¿Qué hace la directiva 'bind *:443 ssl crt /etc/ssl/cert.pem'?

1. Crea un certificado
2. Escucha en HTTPS usando el certificado indicado
3. Redirige el 443
4. Comprueba la validez del PEM

? Pista : Es la sintaxis para activar la terminación SSL en un puerto.

Artículos para profundizar :

Q41 : ¿Qué ventaja tiene 'nbthread' sobre 'nbproc'?

1. Más rápido
2. Memoria compartida entre hilos para las Stick Tables
3. Menos líneas de código
4. Mejor soporte de Windows

? Pista : Los hilos comparten el mismo espacio de memoria, lo que simplifica mucho la sincronización de datos.

Artículos para profundizar :

Q42 : En las estadísticas, ¿qué significa 'Sessions Cur'?

1. Número total de sesiones desde el inicio
2. Número de sesiones activas actualmente
3. Velocidad de las peticiones
4. Número de sesiones perdidas

? Pista : Cur = Current (actual).

Artículos para profundizar :

Q43 : ¿Qué opción envía los logs de HAProxy a un servidor Syslog?

1. log 127.0.0.1 local0
2. syslog-server 127.0.0.1
3. output-logs local0
4. send-logs

? Pista : HAProxy no escribe en archivos: envía a Syslog (a menudo mediante /dev/log).

Artículos para profundizar :

Q44 : ¿Cómo detecta HAProxy que una aplicación devuelve un error 500?

1. Automáticamente
2. Mediante 'http-check expect status 200'
3. Analizando los logs
4. No puede

? Pista : Se pueden indicar los códigos de éxito esperados para validar la salud.

Artículos para profundizar :

Q45 : El ALOHA de HAProxy Technologies es:

1. Una bebida
2. Un appliance (físico o virtual) listo para usar
3. Un nuevo algoritmo
4. Un script de despliegue

? Pista : ALOHA es la solución de balanceador 'plug and play' de HAProxy.

Artículos para profundizar :

Q46 : ¿Qué modo permite usar HAProxy como Ingress Controller de Kubernetes?

1. Modo Standalone
2. HAProxy Ingress Controller
3. Modo Sidecar
4. Modo Gateway

? Pista : Es un componente específico que traduce los recursos Ingress de Kubernetes en configuración de HAProxy.

Artículos para profundizar :

Q47 : ¿Qué hace la directiva 'balance leastconn'?

1. Elige el servidor más barato
2. Elige el servidor con menos conexiones activas
3. Elige el último servidor usado
4. Elige el servidor más rápido

? Pista : Es el algoritmo dinámico más usado para un balanceo de carga inteligente.

Artículos para profundizar :

Q48 : ¿Se puede modificar el peso (weight) de un servidor sin reiniciar?

1. No
2. Sí, mediante la Runtime API
3. Sí, modificando el archivo
4. Solo a partir de la versión 3.0

? Pista : La Runtime API permite modificar los pesos de forma dinámica (por ejemplo para un despliegue Blue/Green).

Artículos para profundizar :

Q49 : ¿Qué directiva activa la monitorización con Prometheus?

1. prometheus-enable
2. http-request use-service prometheus-exporter
3. stats prometheus
4. option metrics

? Pista : *HAProxy expone de forma nativa un exporter de Prometheus mediante esta directiva de servicio.*

Artículos para profundizar :

Q50 : ¿Cuál es la duración por defecto de una versión LTS en HAProxy Technologies?

1. 6 meses
2. 1 año
3. Entre 2 y 5 años
4. Indefinida

? Pista : *Las versiones LTS (Long Term Support) ofrecen varios años de tranquilidad.*

Artículos para profundizar :

Q51 : Una sección 'listen' es útil para:

1. Sustituir a global
2. Combinar frontend y backend en un solo bloque
3. Escuchar música
4. Nada, está obsoleta

? Pista : *Simplifica las configuraciones sencillas (por ejemplo un proxy TCP o las estadísticas).*

Artículos para profundizar :

Q52 : ¿Qué significa 'timeout client 50s'?

1. El cliente debe responder en 50 s
2. Inactividad máxima del cliente antes del corte
3. Tiempo de carga de la página
4. Retardo antes de reintentar

? Pista : *Es el tiempo de inactividad permitido del lado del cliente.*

Artículos para profundizar :

Q53 : ¿Para qué sirve la opción 'redispatch'?

1. Volver a enviar la petición
2. Permitir reenviar una petición a otro servidor si el elegido cae
3. Limpiar la caché
4. Cambiar de algoritmo

? Pista : *Garantiza la continuidad del servicio si un servidor deja de estar disponible justo después de elegirlo.*

Artículos para profundizar :

Q54 : ¿Cómo definir una ACL que compruebe si el usuario usa un móvil?

1. `acl is_mobile user_agent -i mobile`
2. `acl is_mobile hdr_sub(user-agent) -i mobile`
3. `match mobile`
4. `check device mobile`

? Pista : *Se inspecciona la cabecera 'User-Agent' buscando la cadena 'mobile'.*

Artículos para profundizar :

Q55 : La opción '-D' al arrancar HAProxy significa:

1. Debug
2. Daemon (se ejecuta en segundo plano)
3. Delete
4. Default

? Pista : *Imprescindible para que el proceso se ejecute en segundo plano.*

Artículos para profundizar :

Q56 : ¿Qué ocurre si no hay ningún backend disponible?

1. HAProxy se detiene
2. HAProxy devuelve un error 503 Service Unavailable
3. La petición espera indefinidamente
4. El tráfico va al frontend

? Pista : *HAProxy genera él mismo la página de error 503.*

Artículos para profundizar :

Q57 : ¿Qué directiva permite personalizar las páginas de error?

1. error-page
2. errorfile
3. custom-error
4. http-error

? Pista : *'errorfile' apunta a un archivo HTML personalizado para cada código HTTP.*

Artículos para profundizar :

Q58 : La opción 'log-format' permite:

1. Cambiar el color de los logs
2. Personalizar la información de cada línea de log
3. Comprimir los logs
4. Desactivar los logs

? Pista : Fundamental para extraer cabeceras concretas o tiempos de respuesta precisos.

Artículos para profundizar :

Q59 : ¿Qué hace 'balance uri'?

1. Reparte según la URL para optimizar la caché de los backends
2. Reparte al azar
3. Comprueba si la URL es válida
4. Bloquea ciertas URL

? Pista : Ideal para arquitecturas de servidores de caché (como Varnish).

Artículos para profundizar :

Q60 : ¿Qué nombre de cookie se usa por defecto para la persistencia por cookie?

1. PHPSESSID
2. SERVERID
3. SRV
4. HA_COOKIE

? Pista : SERVERID es el nombre convencional que se usa a menudo en la documentación.

Artículos para profundizar :

Q61 : En HAProxy One, ¿qué es Fusion?

1. Un motor de renderizado
2. El plano de control centralizado
3. El nombre del soporte técnico
4. Un módulo de compresión

? Pista : *Fusion permite gestionar y supervisar todo el parque HAProxy Enterprise desde una sola interfaz.*

Artículos para profundizar :

Q62 : ¿Qué significa 'SSL Passthrough'?

1. HAProxy no descifra: transmite el flujo cifrado al backend
2. HAProxy elimina el SSL
3. HAProxy genera un certificado falso
4. El SSL es opcional

? Pista : *HAProxy trabaja entonces en capa 4 (TCP) y no ve el contenido HTTP.*

Artículos para profundizar :

Q63 : ¿Qué directiva permite limitar el caudal (rate limiting)?

1. limit-rate
2. http-request track-sc0 src
3. rate-limit
4. conn-limit

? Pista : *Combinado con las Stick Tables, permite contar y bloquear los excesos de peticiones.*

Artículos para profundizar :**Q64 : El WAF integrado en HAProxy Enterprise sirve para:**

1. Acelerar la red
2. Proteger contra los ataques de aplicación (SQLi, XSS)
3. Gestionar contraseñas
4. Sustituir al antivirus

? Pista : *Web Application Firewall: una barrera de seguridad que filtra el tráfico malicioso.*

Artículos para profundizar :**Q65 : ¿Qué comando muestra la versión de HAProxy?**

1. haproxy -v
2. haproxy --version
3. haproxy version
4. cat /proc/haproxy/version

? Pista : *La opción -v muestra la versión y las opciones de compilación.*

Artículos para profundizar :**Q66 : ¿Para qué sirve 'stats auth admin:password'?**

1. Para crear un usuario del sistema
2. Para proteger con contraseña el acceso al dashboard de estadísticas
3. Para autenticar los servidores
4. Para nada

? Pista : *Seguridad básica para no exponer las estadísticas a todo el mundo.*

Artículos para profundizar :

Q67 : ¿Qué directiva de 'global' reduce los privilegios después del arranque?

1. user / group
2. reduce-privs
3. low-priv
4. chroot

? Pista : HAProxy arranca como root para abrir los puertos (80/443) y después cambia al usuario definido.

Artículos para profundizar :

Q68 : ¿Para qué sirve el 'chroot'?

1. Acelerar el disco
2. Aislar HAProxy en un directorio vacío para limitar el impacto de una vulnerabilidad
3. Cambiar de raíz
4. Optimizar los logs

? Pista : Es una medida de seguridad del sistema (jail).

Artículos para profundizar :

Q69 : ¿Se puede usar HAProxy como proxy de salida (Forward Proxy)?

1. Sí
2. No, es solo un Reverse Proxy
3. Solo en la versión 1.5
4. Solo con Squid

? Pista : HAProxy está diseñado para ser un Reverse Proxy (delante de los servidores).

Artículos para profundizar :

Q70 : ¿Qué directiva activa la compresión GZIP?

1. gzip enable
2. compression algo gzip
3. http-request compress
4. option zip

? Pista : *Permite reducir el ancho de banda consumido por los archivos de texto.*

Artículos para profundizar :

Q71 : Un 'Pod' de Kubernetes puede contener HAProxy mediante:

1. Un Ingress
2. Un Sidecar
3. Un Service
4. Las tres son posibles

? Pista : *HAProxy es muy flexible en las arquitecturas Kubernetes.*

Artículos para profundizar :

Q72 : ¿Cuál de estas afirmaciones sobre HAProxy One es cierta?

1. Es un motor de base de datos
2. Es una plataforma unificada para el ecosistema HAProxy
3. Es una versión gratuita limitada
4. Es un plugin de navegador

? Pista : *HAProxy One unifica Enterprise, Fusion y Edge.*

Artículos para profundizar :

Q73 : La opción 'http-buffer-request' sirve para:

1. Almacenar la petición en memoria
2. Esperar al final del cuerpo antes de enrutar (útil para el WAF)
3. Acelerar la CPU
4. Limitar los logs

? Pista : *Imprescindible para que el WAF pueda inspeccionar un POST completo.*

Artículos para profundizar :

Q74 : ¿Qué hace 'http-request set-header Server MyProxy'?

1. Cambia el nombre del servidor backend
2. Añade o modifica la cabecera HTTP 'Server' de la petición
3. Define el nombre de la instancia
4. Es un error de sintaxis

? Pista : *Todas las cabeceras se pueden manipular al vuelo.*

Artículos para profundizar :

Q75 : El protocolo PROXY (Proxy Protocol) sirve para:

1. Acelerar la red
2. Transmitir la IP real del cliente incluso en modo TCP (capa 4)
3. Autenticar los servidores
4. Sustituir al HTTPS

? Pista : *Inventado por HAProxy, se ha convertido en un estándar para conservar la visibilidad de la IP de origen.*

Artículos para profundizar :

Q76 : ¿Qué versión de HAProxy introdujo el soporte nativo de Prometheus?

1. 1.0
2. 2.0
3. 1.5
4. 2.8

? Pista : *La rama 2.x aportó muchísimas funcionalidades de observabilidad.*

Artículos para profundizar :

Q77 : La directiva 'external-check' permite:

1. Comprobar el DNS
2. Lanzar un script Bash/Python personalizado para probar un servicio complejo
3. Hacer checks desde Internet
4. Comprobar el hardware

? Pista : *Muy útil cuando un simple check TCP o HTTP no basta.*

Artículos para profundizar :

Q78 : En el dashboard de estadísticas, ¿qué significa el color rojo para un servidor?

1. Demasiado tráfico
2. Servidor DOWN (fallo de los health checks)
3. Servidor en mantenimiento
4. Certificado caducado

? Pista : *El rojo indica una avería detectada.*

Artículos para profundizar :

Q79 : ¿Y el color naranja?

1. Sobrecarga
2. Mantenimiento (drain mode)
3. Certificado a punto de caducar
4. Servidor arrancando

? Pista : *Se usa generalmente para el modo Maintenance o Drain.*

Artículos para profundizar :

Q80 : ¿Qué componente de HAProxy Technologies está especializado en absorber ataques DDoS globales?

1. HAProxy OSS
2. HAProxy Fusion
3. HAProxy Edge
4. ALOHA

? Pista : *HAProxy Edge usa su red distribuida para absorber el tráfico malicioso antes de que llegue a tus servidores.*

Artículos para profundizar :

Q81 : ¿Cuántos frontends se pueden definir en un archivo?

1. Solo uno
2. Como máximo 10
3. Ilimitados
4. Tantos como núcleos de CPU

? Pista : *HAProxy es extremadamente escalable.*

Artículos para profundizar :

Q82 : ¿Qué hace 'option abortonclose'?

1. Cierra HAProxy
2. Detiene el tratamiento de la petición si el cliente cierra la conexión antes de tiempo
3. Elimina los logs
4. Ignora los errores

? Pista : *Evita malgastar recursos en los servidores si el cliente ya se ha ido.*

Artículos para profundizar :

Q83 : En una ACL, la palabra clave 'ssl_fc' comprueba:

1. La velocidad SSL
2. Si la conexión actual está cifrada (Front Connection)
3. La validez del certificado
4. El nombre del dominio

? Pista : *SSL Front Connection.*

Artículos para profundizar :

Q84 : ¿Qué directiva permite incluir otro archivo?

1. include
2. import
3. No es posible de forma nativa con una directiva interna
4. f-include

? Pista : *HAProxy usa el argumento de línea de comandos -f para cargar varios archivos; no hay directiva 'include' dentro del .cfg.*

Artículos para profundizar :

Q85 : ¿Qué significa 'L4OK'?

1. El check TCP ha funcionado
2. El check HTTP ha funcionado
3. El servidor es potente
4. Es un error

? Pista : *Layer 4 OK.*

Artículos para profundizar :

Q86 : ¿Para qué sirve un archivo 'Map'?

1. Mostrar un mapa
2. Almacenar miles de correspondencias clave/valor para ACL ultrarrápidas
3. Definir las rutas de red
4. Localizar a los clientes

? Pista : *Es el método eficiente para gestionar un enrutamiento multidominio masivo.*

Artículos para profundizar :

Q87 : La opción 'dontlog-normal' sirve para:

1. No registrar nada
2. Registrar solo los errores (reducir el volumen de logs)
3. Registrar en binario
4. Registrar solo en disco

? Pista : *Útil para ahorrar espacio en disco si todo va bien.*

Artículos para profundizar :

Q88 : ¿Qué hace 'http-request capture req.hdr(Referer) len 64'?

1. Bloquea el Referer
2. Registra el Referer en los logs (captura)
3. Limita el tamaño de las peticiones
4. Cuenta los referers

? Pista : *Permite añadir información específica a los logs de HAProxy.*

Artículos para profundizar :

Q89 : ¿Se pueden hacer SSL Passthrough y terminación SSL en el mismo HAProxy?

1. No
2. Sí, en frontends distintos
3. Solo con Fusion
4. Solo en modo UDP

? Pista : *HAProxy es polivalente.*

Artículos para profundizar :

Q90 : ¿Qué directiva activa el protocolo HTTP/2?

1. option http2
2. alpn h2,http/1.1 (en la línea bind)
3. mode h2
4. http2-enable

? Pista : *ALPN permite que el navegador y HAProxy negocien el paso a HTTP/2.*

Artículos para profundizar :

Q91 : ¿Qué significa 'HAPEE'?

1. Happy HAProxy
2. HAProxy Enterprise Edition
3. HAProxy Enhanced
4. Es una broma

? Pista : *Es el acrónimo que designa la versión Enterprise.*

Artículos para profundizar :

Q92 : ¿Cómo facilita HAProxy One la seguridad?

1. Instalando un antivirus
2. Centralizando las políticas de WAF y de gestión de bots
3. Cifrando el disco duro
4. Limitando el número de frontends

? Pista : *Una plataforma unificada aplica la seguridad de forma coherente en todas partes.*

Artículos para profundizar :

Q93 : La directiva 'maxconn' en una línea server de un backend sirve para:

1. Limitar las conexiones de clientes
2. Limitar el número de peticiones simultáneas enviadas a ESE servidor concreto
3. Definir el ancho de banda
4. Es un error

? Pista : *Crucial para proteger un servidor backend contra la saturación.*

Artículos para profundizar :

Q94 : El modo 'drain' en un servidor significa:

1. El servidor se elimina
2. Ya no se le envían NUEVOS clientes, pero se dejan terminar las sesiones en curso
3. Se vacían sus archivos
4. Es un check de disco

? Pista : *Imprescindible para un mantenimiento limpio sin desconectar a nadie.*

Artículos para profundizar :

Q95 : ¿Cuál es la licencia de HAProxy OSS?

1. Propietaria
2. GPLv2
3. MIT
4. Apache 2.0

? Pista : *HAProxy es un proyecto de código abierto histórico con licencia GPL.*

Artículos para profundizar :

Q96 : El creador de HAProxy es:

1. Linus Torvalds
2. Willy Tarreau
3. Brendan Gregg
4. Solomon Hykes

? Pista : *Willy Tarreau sigue manteniendo activamente el proyecto.*

Artículos para profundizar :

Q97 : En Kubernetes, ¿qué anotación activa HAProxy?

1. kubernetes.io/ingress.class: haproxy
2. proxy: true
3. use-haproxy: yes
4. service.type: lb

? Pista : Es el método estándar para seleccionar el Ingress Controller.

Artículos para profundizar :

Q98 : ¿El ecosistema HAProxy One es solo para el cloud?

1. Sí
2. No, admite on-premise, multicloud e híbrido
3. Solo AWS
4. Solo Azure

? Pista : HAProxy está diseñado para ser independiente de la infraestructura.

Artículos para profundizar :

Q99 : ¿Qué directiva comprime las respuestas?

1. option compress
2. compression algo gzip
3. http-request gzip
4. zip-on

? Pista : Se definen el algoritmo y los tipos MIME que se comprimen.

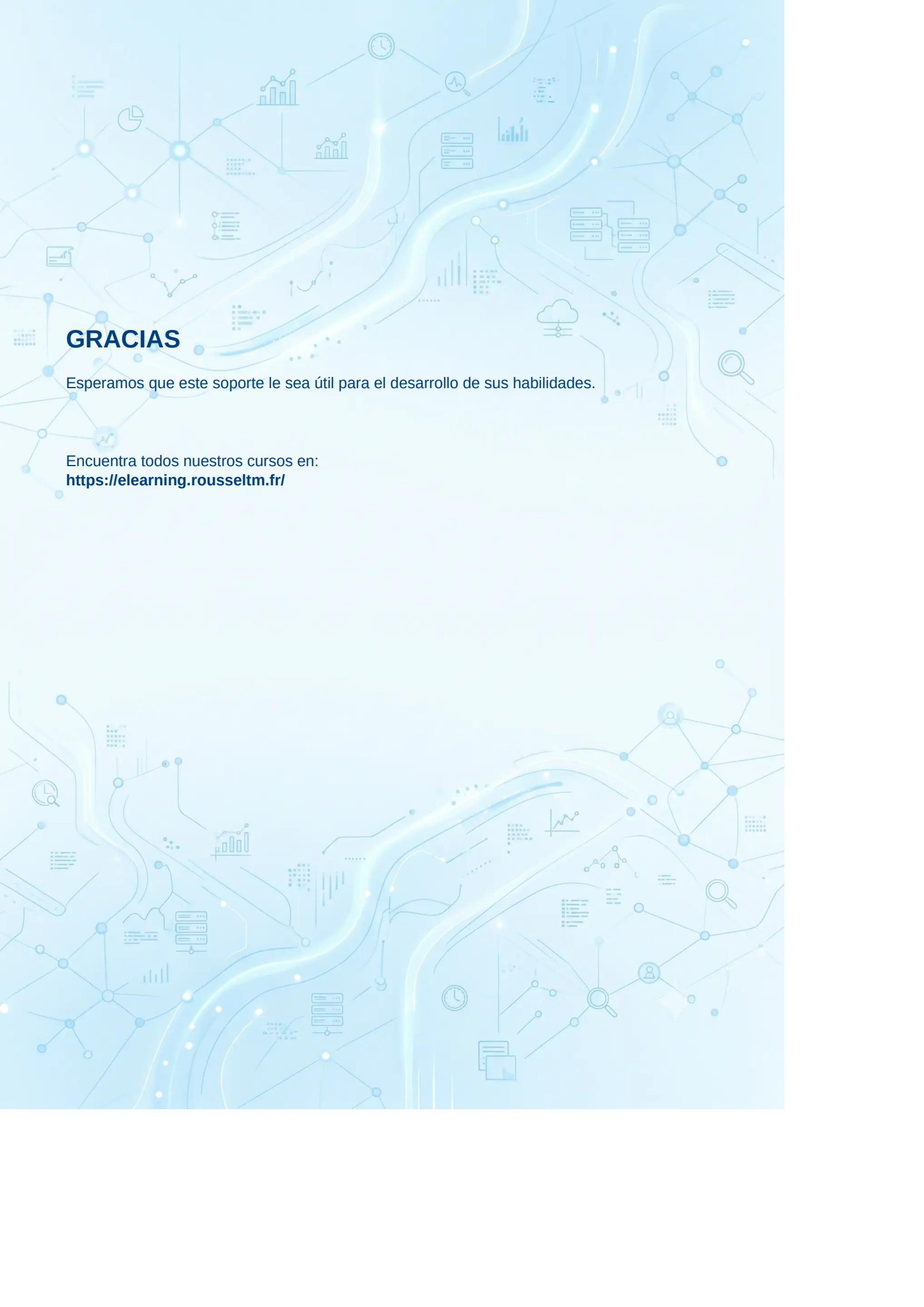
Artículos para profundizar :

Q100 : ¡Enhorabuena, has terminado! ¿Cuál es el último consejo de HAProxy?

1. Prueba siempre tu configuración con -c
2. Usa el puerto 80 para todo
3. No leas los logs
4. Reinicia cada hora

? Pista : *La validación de la configuración es la base de toda puesta en producción tranquila.*

Artículos para profundizar :



GRACIAS

Esperamos que este soporte le sea útil para el desarrollo de sus habilidades.

Encuentra todos nuestros cursos en:
<https://elearning.rousseltm.fr/>