Duración: 2 sesiones

Requisitos Previos:

- Haber completado satisfactoriamente la Práctica 1 o tener experiencia equivalente configurando servidores web con Docker.
- Conocimientos básicos sobre estrategias de balanceo de carga.

La práctica se realizará de manera individual. Tiene un peso del **30%** del total de prácticas.

Introducción

El balanceo de carga es esencial en la implementación de sitios web de alto tráfico, asegurando que ninguna única instancia del servidor web soporte demasiada carga. Esta práctica explora la implementación y configuración de nginx y HAProxy como balanceadores de carga, utilizando distintas estrategias de balanceo y monitoreando su desempeño.

Objetivos de la Práctica:

Esta práctica se centrará en crear un escenario multicontenedor utilizando contenedores Docker y estrategias de balanceo de carga.

1. Aprender la configuración y uso de nginx y HAProxy como balanceadores de carga.
2. Implementar distintas estrategias de balanceo de carga, incluyendo round-robin, entre otras.
3. Configurar y analizar las estadísticas de balanceo de carga para nginx y HAProxy.

Descripción de la Práctica:

Se comenzará por configurar un entorno Docker, donde los balanceadores de carga distribuyan de manera eficiente las solicitudes entrantes a un conjunto de servidores web, garantizando así una alta disponibilidad y escalabilidad de la aplicación. Se iniciará la práctica estableciendo una red Docker de tipo bridge, denominada red_web, con una configuración de red especificada para tener una dirección IP de 192.168.10.50. En este entorno, se prepararán al menos 2 contenedores que funcionarán como balanceadores de carga. Cada uno de estos contenedores estará configurado con software de balanceo de carga específico: uno con nginx y otro con HAProxy.

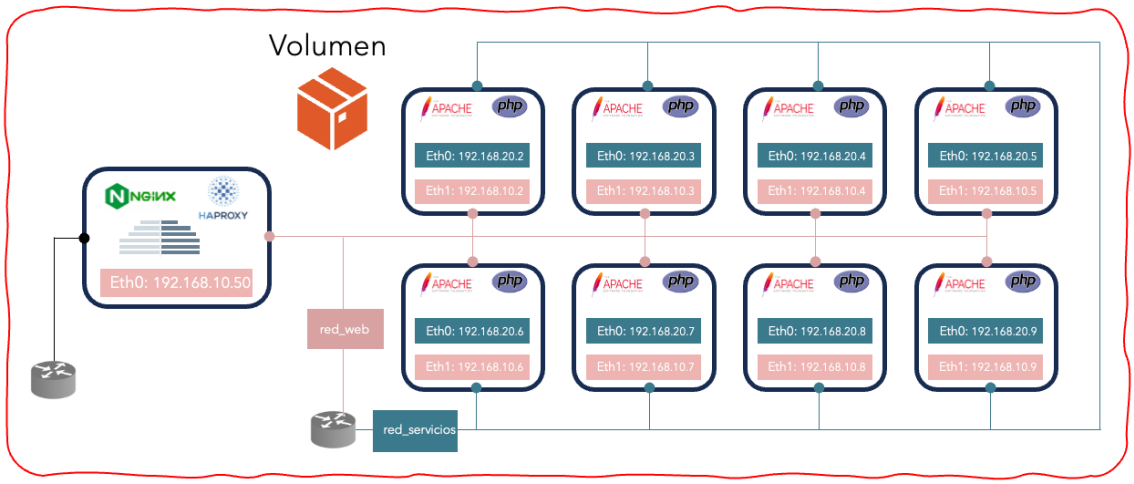
Los contenedores de balanceo de carga se conectarán a la red **red_web**. Esta configuración asegura que los balanceadores de carga puedan comunicarse eficientemente con los contenedores de los servidores web situados dentro de la misma red. Se establecerá una dependencia directa entre los balanceadores de carga y 8 contenedores de servidores web previamente configurados. Esto se hace para asegurar que los balanceadores de carga sólo comiencen a distribuir solicitudes una vez que todos los servidores web estén operativos y listos para manejar el tráfico.

Dentro de la configuración de nginx y HAProxy, se implementarán diversas estrategias de balanceo, incluyendo pero no limitándose a round-robin. Estas estrategias permitirán distribuir de manera equitativa las solicitudes entrantes entre los servidores web disponibles. Se prestará especial atención a la configuración de las estadísticas

de balanceo en ambos balanceadores. Esto incluirá habilitar páginas de estadísticas específicas que proporcionen información detallada sobre el número de solicitudes, su distribución entre los servidores y la salud general de la granja web.

Para finalizar, tras la configuración, se realizarán pruebas exhaustivas para verificar la eficacia del balanceo de carga. Esto implicará enviar solicitudes a la dirección IP de los balanceadores de carga y observar la distribución de estas solicitudes entre los servidores web. Se accederá a las interfaces de estadísticas de nginx y HAProxy para evaluar el rendimiento de los balanceadores de carga, la distribución de solicitudes y cualquier posible punto de mejora.

Esquema general de la práctica:



Desarrollo:

Se pretende desarrollar y configurar un entorno de balanceo de carga utilizando dos de los balanceadores más populares: nginx y HAProxy. Se crearán, al menos 2 contenedores para cada balanceador de carga, los cuales estarán conectados a una red específica denominada red_web, con la dirección IP 192.168.10.50. A través de esta configuración, se establecerán dependencias con los 8 contenedores de servidores web desplegados en la Práctica 1, permitiendo experimentar con diversas estrategias de balanceo, como round-robin, entre otras. Además, se configurarán y analizarán las estadísticas de balanceo de carga de cada balanceador.

Parte 0: Creación del directorio y espacio de trabajo para cada balanceador

En esta parte se establecerá el espacio de trabajo a través de directorios específicos donde se crearán los archivos de configuración, así como la web del escenario.

- Crea un directorio en tu máquina local llamado P2-tuusuariougr-nginx para trabajar con los archivos de configuración de ningx y otro directorio llamado P2-tuusuariougr-haproxy para trabajar con los archivos de configuración de haproxy.
- Copia en cada directorio de los balanceadores, es decir en P2-tuusuariougr-nginx y P2-tuusuariougr-haproxy, el directorio que creaste en la práctica 1 llamado web_tuusuariougr para que los servidores web sirvan el index.php que creaste en la práctica 1.

Parte 1: Configuración de Nginx como Balanceador de Carga

En esta parte configuraremos los archivos de configuración de nginx así como crearemos imágenes personalizadas con Dockerfile para la imagen de nginx.

- Dockerfile para Nginx:
  - En el directorio P2-tuusuariougr-nginx, crea un archivo llamado DockerfileNginx para crear una imagen a partir de la imagen oficial de nginx: `FROM nginx:latest`

- Archivo de Configuración de Nginx (nginx.conf):
  - Crea un archivo nginx.conf en el directorio P2-tuusuariougr-nginx con la configuración de balanceo de carga. A continuación, se detalla un ejemplo de configuración básica aplicando round-robin como algoritmo de balanceo y registro de accesos y errores, y habilitadas las estadísticas.

1.1: Configuración de Nginx con Docker Compose

En esta parte se configurará Nginx como balanceador de carga para los servidores web Apache utilizando Docker Compose, con conexiones a las redes red_web y red_servicios y se establecerán dependencias entre los servicios.

- Construcción de Servidores Web Apache:
  - Utilizar el DockerfileApache de la Práctica 1 para construir los contenedores de Apache con el nuevo tag p2.
- Definición de Servicios en docker-compose.yml:
  - Definir un servicio para cada instancia de Apache con las siguientes características:
    - Imagen construida a partir del DockerfileApache de la Práctica 1.
    - Nombre de la imagen personalizada como tuusuarioUGR-apache-image:p2.
    - Nombre del contendedor: webX donde X es el número de contenedor del 1 al 8.
    - Volumen para montar el directorio local web_tuusuarioUGR en la ruta por defecto de Apache para servir el index.php.
    - Conexión a las redes red_web y red_servicios.

- Definir un servicio llamado balanceador-nginx que incluya:
  - Construcción de la imagen a partir del DockerFileNginx.
  - Nombre de la imagen personalizada como tuusuarioUGR-nginx-image:p2.
  - Volumen para montar el archivo nginx.conf en el contenedor en /etc/nginx/nginx.conf.
  - Asignación de una dirección IP estática 192.168.10.50 en la red red_web.
  - Dependencia establecida con los servicios de Apache para garantizar el orden correcto de despliegue.

NOTA: Si declaras una red como "external" en el archivo docker-compose.yml, le estás indicando a Docker Compose que no gestione la red (ni la creación ni la eliminación) porque es una red creada fuera de Docker Compose, y deberías haberla creado manualmente con anterioridad usando "docker network create" o debería estar siendo gestionada por otro proyecto de Docker Compose.

1.2: Verificación y Pruebas del balanceador nginx

En esta parte realizaremos el despliegue del escenario con el balanceador nginx y los servidores web finales, así como verificaremos el correcto funcionamiento del escenario.

- Despliegue y Ejecución:
  - Ejecutar docker-compose up -d para iniciar todos los servicios definidos en el archivo.
  - Verificar que los servicios están corriendo y que Nginx está balanceando las cargas como se espera.

- Verificación y Pruebas:
  - Comprobar que Nginx distribuye correctamente las solicitudes entre los distintos contenedores de Apache.

  - Acceder a la página de estadísticas de nginx que hemos configurado previamente (/estadisticas_tuusuarioUGR) para observar el rendimiento del balanceador.

Parte 2: Configuración de HAProxy como Balanceador de Carga

En esta sección, configuraremos HAProxy para que funcione como balanceador de carga, estableciendo su configuración y preparando una imagen personalizada con Dockerfile.

- Dockerfile para HAProxy:
  - Dentro del directorio P2-tuusuariougr-haproxy, crea un archivo llamado DockerfileHAProxy.
  - Usa la imagen oficial de HAProxy como punto de partida: FROM haproxy:latest.
  - Agrega instrucciones para copiar la configuración personalizada de HAProxy al contenedor.

- Archivo de Configuración de HAProxy (haproxy.cfg):
  - Crea un archivo haproxy.cfg en el directorio P2-tuusuariougr-haproxy donde añadir las configuraciones necesarias.
  - Configura HAProxy para utilizar algoritmos de balanceo de carga, como round-robin. Ejemplo básico de configuración:

```
frontend frontend usuarioUGR
     bind *:80
     default_backend backend_usuarioUGR

backend backend_usuarioUGR
     server web1 192.168.10.2:80 maxconn 32
     server web2 192.168.10.3:80 maxconn 32
        etc...
```

  - Activa las estadísticas de HAProxy para facilitar el monitoreo de su rendimiento. Ejemplo básico de configuración de estadísticas en el puerto 9000:

```
global
stats socket /var/lib/haproxy/stats
listen stats
         bind *:9000
         mode http
         stats enable
         stats uri /estadisticas_tuusuarioUGR
         stats realm HAProxy\ Statistics
         stats auth tuusuarioUGR:SWAP1234
```

    NOTA: No olvides personalizar los backend, estadísticas, etc. con tuusuarioUGR. Es decir, incluir tu usuario de la UGR en las configuraciones.

2.1: Configuración de HAProxy con Docker Compose

Aquí estableceremos HAProxy como balanceador de carga para los servidores web Apache usando Docker Compose, definiendo las conexiones de red y las dependencias de servicios.

- Construcción de Servidores Web Apache:
  - Utiliza el DockerfileApache de la Práctica 1 para construir las imágenes de los contenedores Apache con el nuevo tag p2.

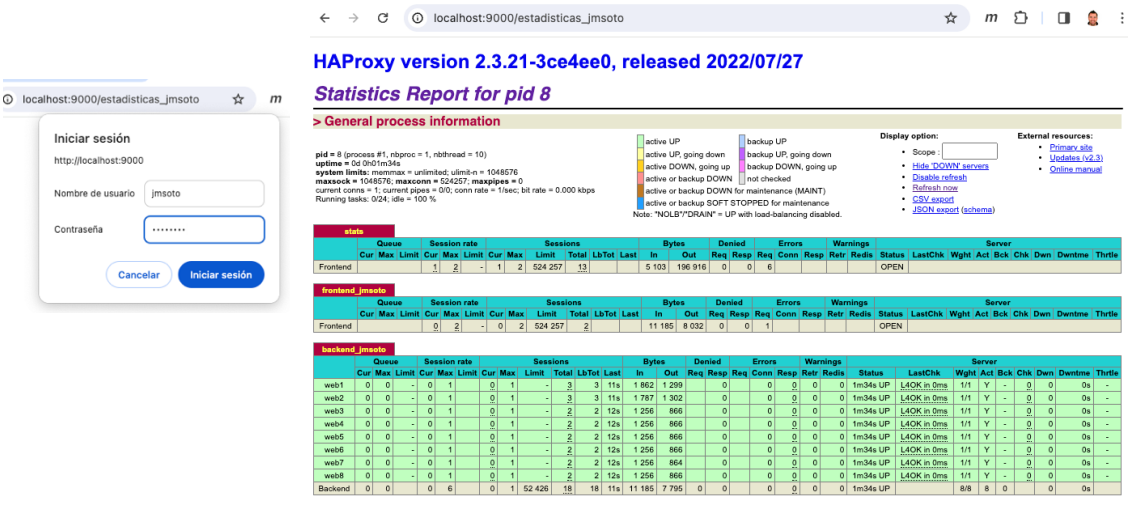
- Definición de Servicios en docker-compose.yml:
  - Define un servicio para cada instancia de Apache que incluya:
    - Usa la imagen construida a partir del **DockerfileApache** de la Práctica 1.
    - Personaliza el nombre de la imagen como tuusuarioUGR-apache-image:p2.
    - Asigna nombres a los contenedores como **webX**, donde **X** es el número de contenedor del 1 al 8.
    - Monta el directorio **web_tuusuarioUGR** local en la ruta por defecto de Apache para servir el **index.php**.
    - Conecta los contenedores a las redes **red_web** y **red_servicios**.
  - Define un servicio llamado **balanceador-haproxy** que incluya:
    - Construcción de la imagen a partir del **DockerfileHAProxy**.
    - Personaliza el nombre de la imagen como tuusuarioUGR-haproxy-image:p2.
    - Monta el archivo **haproxy.cfg** en el contenedor en **/usr/local/etc/haproxy/haproxy.cfg**.
    - Asigna la dirección IP estática **192.168.10.50** en la red **red_web**.
    - Establece la dependencia con los servicios de Apache para garantizar el correcto despliegue secuencial.

*Nota sobre las redes Docker Compose: Como en la configuración de Nginx, si las redes ya están creadas externamente, marca las redes **red_web** y **red_servicios** como externas en tu **docker-compose.yml** para evitar conflictos o intentos de recreación.*

2.2: Verificación y Pruebas del balanceador HAProxy

En esta sección, implementarás el escenario con HAProxy y lo verificarás.

- Despliegue y Ejecución:
  - Ejecuta **docker-compose up -d** para arrancar todos los servicios definidos en el archivo **docker-compose.yml**.
  - Confirma que los servicios están activos y que HAProxy está gestionando la carga efectivamente.
- Verificación y Pruebas:
  - Verifica que HAProxy distribuya adecuadamente las solicitudes entre los diferentes contenedores Apache.
  - Accede a la interfaz de estadísticas (ruta **/ estadisticas_tuusuarioUGR**) para monitorear el rendimiento y la distribución de la carga.



Evaluación

La práctica se realizará de manera individual. Tiene un peso del **30%** del total de prácticas.

Para superar la práctica se deben realizar las siguientes **tareas básicas**:

B1: Preparación del Entorno de Trabajo
- Crear directorios específicos para los archivos de configuración de los balanceadores:
  - P2-tuusuariougr-nginx para nginx.
  - P2-tuusuariougr-haproxy para HAProxy.

B2: Configuración de Nginx como Balanceador de Carga
- Redactar el Dockerfile para crear una imagen personalizada de Nginx a partir de la imagen oficial.
- Escribir el archivo de configuración nginx.conf con la estrategia de balanceo round-robin y configuraciones de registro de accesos y errores.

B3: Implementación del escenario de Ningx con Docker Compose
- Reutilizar o adaptar el DockerfileApache de la Práctica 1 para los contenedores de Apache con el nuevo tag p2.
- Desarrollar el docker-compose.yml para configurar el servicio para cada contenedor Apache, el volumen para el directorio web_tuusuariougr, y para el balanceador de carga Nginx con las características correspondientes.

B4: Verificación y Pruebas del escenario de Nginx
- Desplegar los servicios con docker-compose up -d.
- Verificar que los servicios están activos y que Nginx distribuye correctamente las solicitudes.
- Acceder a la página de estadísticas de Nginx para observar el rendimiento del balanceador.

B5: Configuración de HAProxy como Balanceador de Carga
- Redactar el Dockerfile para crear una imagen de HAProxy a partir de la imagen oficial.
- Crear el archivo de configuración haproxy.cfg incluyendo las estrategias de balanceo de carga y la configuración de las estadísticas.

B6: Implementación del escenario de HAProxy con Docker Compose
- Reutilizar o adaptar el DockerfileApache de la Práctica 1 para los contenedores de Apache con el nuevo tag p2.
- Detallar en docker-compose.yml la configuración de cada servicio Apache, el volumen para el directorio web_tuusuariougr, y el servicio para el balanceador de carga HAProxy.

B7: Verificación y Pruebas del escenario de HAProxy
- Iniciar los servicios con docker-compose up -d asegurando que HAProxy esté operativo.
- Comprobar la correcta distribución de solicitudes por parte de HAProxy.
- Observar el rendimiento a través de la interfaz de estadísticas configurada.

Se proponen, opcionalmente, las siguientes tareas avanzadas:

A1: Configuraciones Avanzadas de Nginx

- Implementar estrategias de balanceo de carga avanzadas en nginx.conf, como el balanceo basado en menor tiempo de respuesta o ponderado y analiza el impacto de esas configuraciones en el escenario de balanceo.

A2: Configuraciones Avanzadas de HAProxy

- Implementar estrategias de balanceo de carga avanzadas en haproxy.cfg, como el balanceo basado en menor tiempo de respuesta o ponderado y analiza el impacto de esas configuraciones en el escenario de balanceo.

A3: Experimentación con Diferentes Balanceadores de Carga

- Configurar y desplegar otros balanceadores de carga disponibles en el ecosistema Docker, como Traefik, gobetween o balanceadores personalizados.

A4: Investigación y Pruebas de Tolerancia a Fallos

- Realizar pruebas de tolerancia a fallos apagando intencionadamente instancias de servidor web para observar la reacción y la reasignación de carga de los balanceadores.

A5: Automatización de escalado del escenario

- Implementar una lógica de escalado automático para añadir o eliminar instancias de servidor web basadas en la carga, horarios específicos o consumo de recursos utilizando Docker Compose. Esta idea podría realizarse mediante:
  1. Monitorización de la Carga de los Servidores:
     - Configura métricas de monitoreo (CPU, memoria, conexiones activas, etc.). para ver el estado de cada contenedor de servidor web. Puedes usar herramientas como cAdvisor, Node Exporter o agentes personalizados que expongan las métricas a un sistema de monitoreo.
  2. Script para escalar servicios:
     - Escribe un script que utilice la API de Docker para obtener las métricas y ajuste la configuración del balanceador de carga en función de estas métricas. Este script podría ajustar el archivo **nginx.conf** o **haproxy.cfg** para añadir o eliminar servidores del backend en el balanceador y luego recargar la configuración del balanceador sin interrumpir el tráfico.
  3. Automatización del script:
     - Configura un trabajo cron que ejecute el script a intervalos regulares, por ejemplo, cada 2 minutos.

  NOTA: Recuerda que la implementación real de este sistema sería bastante compleja y requeriría pruebas exhaustivas. En entornos de producción, es recomendado utilizar plataformas como Kubernetes diseñadas para orquestación que ofrecen funcionalidades de autoescalado y autohealing integradas.

Normas de entrega

Se desarrollará un documento siguiendo el guion de la práctica y **detallando** e indicando, en su caso, los **aspectos básicos y avanzados realizados**, comandos de terminal ejecutados, resultados de ejecución, etc.

- Por ejemplo, si se ha realizado la tarea básica de configuración del entorno, el documento .pdf con la memoria de prácticas debe aparecer una sección titulada: *Tareas Básicas - B1. Configuración del Entorno* donde aparezcan detalladas las configuraciones. De igual forma, si por ejemplo, se han realizado tareas avanzadas de Nginx, *Tareas Avanzadas – A1. Configuraciones Avanzadas con Nginx,* detalles de las configuraciones, explicaciones sobre ellas y explicaciones sobre ellas.

Se recomienda utilizar herramientas de control de Tiempo (por ejemplo, clockify) para contabilizar el tiempo de dedicado a la realización de la práctica.

Se deja a **libre elección** la **estructura y formato** del documento el cual reflejará el correcto desarrollo de la práctica a modo de diario/tutorial siguiendo los puntos descritos anteriormente. Asimismo, se recomienda incluir capturas de pantalla que reflejen el correcto desarrollo de los distintos apartados de la práctica. La **primera página** del documento debe incluir, al menos, **nombre**, **apellidos** y **tiempo dedicado a la práctica** medido con herramientas de control de tiempo.

Para la entrega se habilitará una tarea en PRADO cuya entrega debe seguir OBLIGATORIAMENTE el formato especificado.

1. Un archivo .pdf con el documento desarrollado siguiendo el formato **ApellidosNombreP2.pdf**
2. Un archivo .zip con los distintos archivos de configuraciones, carpetas, etc. necesarios para la ejecución de la práctica siguiendo el formato **ApellidosNombreP2.zip**

Uso de Inteligencia Artificial Generativa.

Para cada práctica es OBLIGATORIO usar herramientas de IA generativa (ChatGPT, Copilot u otras) e incluir enlace al chat/prompt utilizado. También se debe analizar y justificar el resultado que proporciona la herramienta con el resultado final que opta el estudiante para la práctica.

Es OBLIGATORIO incluir en el guion una sección titulada: "**Análisis propuesta IA**" donde se incluya enlace al chat/prompt con las consulta/as realizada/as, resultado que proporciona la IA y un párrafo con un análisis crítico y detallado del resultado proporcionado.

Evaluación

La práctica se evaluará mediante el uso de rúbrica específica (accesible por el estudiante en la tarea de entrega) y una defensa final de prácticas.

La detección de prácticas copiadas implicará el suspenso inmediato de todos los implicados en la copia (tanto del autor del original como de quien las copió). OBLIGATORIO ACEPTAR LICENCIA EULA DE TURNITIN en la entrega. Si la memoria supera un 40% de copia Turnitin implicará el suspenso automáticamente.

Rúbrica

Criterios de Evaluación	Excelente (Puntuación máxima)	Bueno (75% de la puntuación máxima)	Adecuado (50% de la puntuación máxima)	Deficiente (25% de la puntuación máxima)	No Realizado (0 puntos)
Tareas básicas					
B1. Preparación del Entorno de Trabajo (5 puntos)	Entorno configurado con directorios correctamente creados y documentados. **(5 puntos)**	Entorno correctamente configurado con pequeños errores en documentación. **(3.75 puntos)**	Entorno parcialmente configurado o con errores significativos. **(2.5 puntos)**	Entorno configurado incorrectamente o con errores graves. **(1.25 puntos)**	No realizado. **(0 puntos)**
B2. Configuración de Nginx (15 puntos)	Dockerfile y nginx.conf perfectamente configurados y documentados. **(15 puntos)**	Configuraciones generalmente correctas con menores errores. **(11.25 puntos)**	Configuraciones con errores o falta de optimización. **(7.5 puntos)**	Configuraciones incorrectas que afectan la funcionalidad. **(3.75 puntos)**	No realizado. **(0 puntos)**
B3. Docker Compose de Nginx (30 puntos)	docker-compose.yml para Nginx configurado y funcionando sin errores. **(30 puntos)**	Configuración con pequeñas inexactitudes pero funcional. **(22.5 puntos)**	Errores que afectan parcialmente la funcionalidad. **(15 puntos)**	Errores graves que impiden el correcto funcionamiento. **(7.5 puntos)**	No realizado. **(0 puntos)**
B4. Verificación de Nginx (5 puntos)	Servicios de Nginx desplegados y verificados, estadísticas revisadas. **(5 puntos)**	Mayoría verificada, con pequeños errores en las pruebas. **(3.75 puntos)**	Algunos servicios verificados, errores en estadísticas. **(2.5 puntos)**	Intento de verificación pero con errores graves. **(1.25 puntos)**	No realizado. **(0 puntos)**
B5. Configuración de HAProxy (15 puntos)	Dockerfile y haproxy.cfg configurados sin errores. **(15 puntos)**	Configuraciones generalmente correctas con menores errores. **(11.25 puntos)**	Configuraciones con errores o falta de optimización. **(7.5 puntos)**	Configuraciones incorrectas que afectan la funcionalidad. **(3.75 puntos)**	No realizado. **(0 puntos)**
B6. Docker Compose de HAProxy (30 puntos)	docker-compose.yml para HAProxy configurado y funcionando sin errores. **(30 puntos)**	Configuración con pequeñas inexactitudes pero funcional. **(22.5 puntos)**	Errores que afectan parcialmente la funcionalidad. **(15 puntos)**	Errores graves que impiden el correcto funcionamiento. **(7.5 puntos)**	No realizado. **(0 puntos)**
B7. Verificación de HAProxy (5 puntos)	Servicios de HAProxy desplegados y verificados, estadísticas revisadas. **(5 puntos)**	Mayoría verificada, con pequeños errores en las pruebas. **(3.75 puntos)**	Algunos servicios verificados, errores en estadísticas. **(2.5 puntos)**	Intento de verificación pero con errores graves. **(1.25 puntos)**	No realizado. **(0 puntos)**

Tareas avanzadas					
A1. Configuraciones Avanzadas de Nginx (10 puntos)	Implementación y documentación de estrategias de balanceo avanzadas en nginx.conf, optimizadas y evaluadas correctamente. **(10 puntos)**	Estrategias de balanceo avanzadas implementadas con pequeños errores o falta de optimización. **(7.5 puntos)**	Estrategias de balanceo básicas implementadas sin ajustes avanzados. **(5 puntos)**	Intento de implementación pero con errores significativos en el balanceo o la configuración. **(2.5 puntos)**	No realizado. **(0 puntos)**
A2. Configuraciones Avanzadas de HAProxy (10 puntos)	Estrategias de balanceo avanzadas en haproxy.cfg implementadas, documentadas y analizadas con precisión. **(10 puntos)**	Estrategias de balanceo implementadas con inexactitudes menores o falta de análisis detallado. **(7.5 puntos)**	Estrategias de balanceo básicas configuradas sin análisis de impacto. **(5 puntos)**	Configuración con errores que afectan la eficacia del balanceo. **(2.5 puntos)**	No realizado. **(0 puntos)**
A3. Experimentación con Diferentes Balanceadores de Carga (30 puntos)	Configuración exitosa y documentada de otros (al menos 2) balanceadores de carga, con análisis comparativo detallado. **(30 puntos)**	Uso adecuado de otros balanceadores con documentación parcial o falta de comparativa. **(22.5 puntos)**	Configuración básica de otros balanceadores sin análisis profundo. **(15 puntos)**	Intento de configuración con errores significativos y sin comparativa. **(7.5 puntos)**	No realizado. **(0 puntos)**
A4. Investigación y Pruebas de Tolerancia a Fallos (10 puntos)	Pruebas de tolerancia a fallos bien ejecutadas, con documentación detallada de la respuesta del sistema. **(10 puntos)**	Pruebas adecuadas de tolerancia a fallos con documentación incompleta. **(7.5 puntos)**	Pruebas básicas de tolerancia a fallos realizadas sin detalles exhaustivos. **(5 puntos)**	Intento de pruebas de tolerancia a fallos pero con graves deficiencias. **(2.5 puntos)**	No realizado. **(0 puntos)**
A5. Automatización de escalado del escenario (40 puntos)	Completa implementación de scripts de automatización con lógica de escalado efectiva y monitorización integrada. **(40 puntos)**	Implementación de escalado con monitorización básica y pequeños errores en la lógica. **(30 puntos)**	Uso de scripts para escalado manual o automatización parcial. **(20 puntos)**	Intento de automatización pero con resultados ineficaces y errores significativos. **(10 puntos)**	No realizado. **(0 puntos)**

Documentación					
Documentación de la Práctica (20 puntos)	Memoria detallada, cuidada, bien estructurada, sin errores ortográficos. **(20 puntos)**	Memoria completa con algunos detalles menores faltantes o errores leves. **(15 puntos)**	Memoria completa pero con varias omisiones o errores ortográficos. **(10 puntos)**	Memoria incompleta, desorganizada o con numerosos errores. **(5 puntos)**	No realizada. **(0 puntos)**
Análisis y Justificación de Resultados de IA Generativa (40 puntos)	Análisis profundo y justificación detallada de las configuraciones y resultados de IA. **(40 puntos)**	Análisis adecuado con justificaciones superficiales o incompletas. **(30 puntos)**	Análisis básico con pocas justificaciones o reflexiones críticas. **(20 puntos)**	Intento de análisis pero con justificaciones inadecuadas. **(10 puntos)**	No realizado. **(0 puntos)**
Comentarios en Configuraciones (15 puntos)	Comentarios detallados y claros en todas las configuraciones. **(15 puntos)**	Comentarios adecuados en la mayoría de las configuraciones. **(11.25 puntos)**	Algunos comentarios útiles, pero falta de detalle o claridad. **(7.5 puntos)**	Comentarios escasos o poco claros. **(3.75 puntos)**	Sin comentarios en las configuraciones. **(0 puntos)**
Penalizaciones					
Requisitos de entrega	Cumple: 0 puntos		No cumple: -20 puntos		
Entrega en plazo fijado	En plazo: 0 puntos		Un poco tarde (horas): -10 puntos	Algo tarde (1 día): -15 puntos	Muy tarde (varios días): -20 puntos
Porcentaje de copia en Turnitin	1-10%: 0 puntos	11-20%: -20 puntos	21-30%: -30 puntos	31-40%: -40 puntos	Más del 40%: Suspenso automático