Duración: 2 sesiones

Requisitos Previos:

- Haber completado satisfactoriamente las Prácticas 1, 2, 3 y 4 o tener experiencia equivalente en configuración de balanceadores de carga, certificados SSL y cortafuegos con Docker.
- Conocimientos básicos sobre benchmarking.

Introducción

Esta práctica tiene como objetivo evaluar el rendimiento y la robustez de la configuración de seguridad y optimización realizada en las prácticas anteriores. Se emplearán dos herramientas de benchmarking: Apache Benchmark (ab) para pruebas básicas sin interfaz y Locust para pruebas de carga avanzadas con interfaz web. Se busca entender el impacto de las configuraciones previas en la capacidad de respuesta y la estabilidad de la granja web frente a diversas cargas de tráfico.

Objetivos de la Práctica:

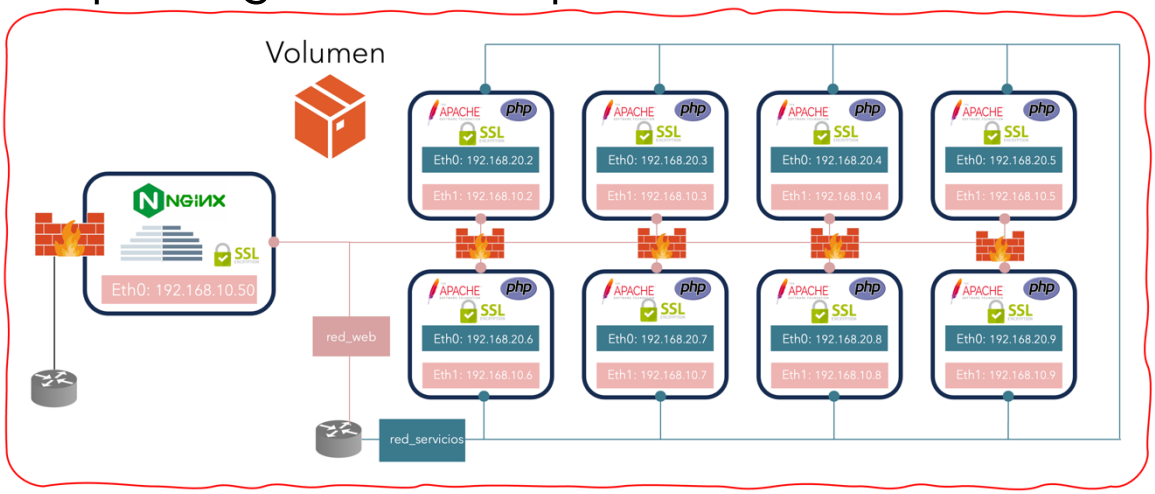
Esta práctica tiene como objetivo principal evaluar el rendimiento de nuestra infraestructura web utilizando contenedores Docker y aplicando diferentes test de carga a través de diferentes herramientas de Benchmarking.

1. Configurar y utilizar Apache Benchmark para realizar pruebas básicas de carga en la granja web.
2. Implementar Locust en un escenario de múltiples contenedores para simular tráfico de usuarios y evaluar el rendimiento del sistema bajo carga.
3. Analizar y comparar los resultados de las pruebas para determinar la eficacia de las configuraciones de seguridad y rendimiento implementadas en prácticas anteriores.

Descripción de la Práctica:

Se profundizará en el análisis y mejora del rendimiento de nuestra granja web mediante la implementación de pruebas de benchmarking. Se emplearán, al menos, dos herramientas distintas para evaluar cómo las configuraciones anteriores afectan la eficiencia y la resistencia del sistema bajo diferentes condiciones de carga. Inicialmente, se utilizará Apache Benchmark (ab), una herramienta de línea de comandos sin interfaz gráfica, para realizar pruebas de rendimiento rápidas y directas que evaluarán la capacidad de respuesta del servidor bajo cargas de tráfico HTTP y HTTPS. Seguidamente, Locust, una herramienta avanzada con interfaz web, permitirá simular comportamientos de usuarios múltiples interactuando con la aplicación en tiempo real, lo que proporcionará una visión más profunda de la capacidad del sistema para manejar tráfico web simultáneo y complejo. Este escenario dual ofrecerá una perspectiva completa sobre cómo los servidores y el balanceador de carga manejan las solicitudes, resaltando tanto las fortalezas como las áreas de mejora potencial. A través de estos análisis, se buscará optimizar la configuración para alcanzar un equilibrio entre seguridad y rendimiento, preparando así nuestra infraestructura para manejar eficazmente las demandas del mundo real.

Esquema general de la práctica:



Desarrollo:

Se establecerá un entorno multicontenedor basado en la granja web configurada en las prácticas anteriores, donde se integrarán herramientas de benchmarking para evaluar el rendimiento bajo diversas condiciones. Se crearán dos escenarios distintos: uno utilizando Apache Benchmark (ab) para lanzar rápidamente peticiones HTTP y HTTPS hacia el balanceador de carga, y otro utilizando Locust con una configuración de un nodo maestro y varios trabajadores para simular tráfico de usuarios en tiempo real.

Parte 0: Creación del espacio de trabajo Benchmarking

En esta parte se establecerá el espacio de trabajo a través de directorios específicos para el conjunto de reglas y script necesarios para la configuración de cortafuegos. Partiendo de la estructura de directorios de la práctica anterior:

- Crea un directorio en tu máquina local llamado **P5-granjaweb** que incluirá los archivos de la práctica anterior, es decir, un Docker-compose-yml para establecer el entorno multicontenedor de los servidores web apache (P4-tuusuariougr-apache), los certificados (P4-tuusuariougr-certificados) y el balanceador (P4-tuusuariougr-nginx) junto con el directorio web_tuusuariougr con el `index.php`.
- Crea un directorio en tu máquina local llamado **P5-ab** donde se incluirán las distintas configuraciones de Apache Benchmark.
- Crea un directorio en tu máquina local llamado **P5-locust** donde se incluirán las distintas configuraciones de Locust.

NOTA: Cada directorio (P5-granjaweb, P5-ab y P5-locust) debe incluir un archivo docker-compose.yml para desplegar el escenario correspondiente.

Parte 1: Implementación de Apache Benchmark

En esta sección, se configurará y utilizará Apache Benchmark (ab) para lanzar pruebas de rendimiento contra el balanceador de carga de la granja web. El objetivo es simular cargas de tráfico HTTP y HTTPS y evaluar cómo la infraestructura responde bajo carga. Para ello, se preparará un contenedor Docker específico que ejecute Apache Benchmark, permitiendo realizar pruebas automatizadas y repetibles. Trabajaremos en el directorio **P5-ab.**

Dockerfile para Apache Benchmark (DockerFileAB):

1. Creación de la Imagen:

El DockerFileAB comenzará con una imagen base de Debian para asegurar un entorno ligero y controlado. Se instalará Apache Benchmark utilizando el gestor de paquetes apt-get, asegurando que todas las dependencias necesarias estén presentes.

Ejemplo básico de Dockerfile

```
FROM debian:latest
RUN apt-get update && apt-get install -y apache2-utils
```

2. Configuración del Contenedor:

El Dockerfile configurará el entorno necesario para ejecutar las pruebas, incluyendo cualquier variable de entorno o configuración de red necesaria. Se asegurará de que el contenedor pueda ejecutar comandos ab sin restricciones adicionales.

Integración con Docker Compose:

- **Definición del Servicio**:

En el D3ocker-compose.yml, se definirá un servicio llamado **apache-benchmark-P5** para el contenedor de Apache Benchmark a partir de la imagen del DockerFileAB que se llamará **tuusuariougr-ab-image:p5**. Este servicio estará configurado para comunicarse con el balanceador de carga dentro de la red_web y tendrá dirección IP 192.168.10.60.

Se establecerán comandos específicos para lanzar peticiones tanto HTTP como HTTPS al balanceador, apuntando a la dirección IP 192.168.10.50 que corresponde al balanceador dentro de la red virtual.

- **Configuración de Red**:

Se asegurará de que el servicio apache-benchmark-P5 se una a la misma red red_web utilizada por la granja web y el balanceador.

Ejemplo básico de Docker-compose.yml

```
services:
  apache-benchmark-P5:
    build:
      context: .
      dockerfile: DockerFileAb
    image: tuusuario-ab-image:p5
    container_name: apache_benchmark-P5
    command: ["ab", "-n", "10000", "-c", "100", "https://192.168.10.50:443/"]
    networks:
      red_web:
        ipv4_address: 192.168.10.60

networks:
  red_web:
    external: true
```

Parte 2: Implementación de pruebas de carga con Locust

En esta parte configuraremos Locust, software de pruebas de carga con una interfaz web, y realizaremos test de carga para simular usuarios interactuando con la granja web bajo condiciones de carga controladas. Trabajaremos en el directorio **P5-locust.**

2.1 Docker Compose para Locust

Definición del Servicio Master:

- Se creará un servicio **master-tuusuariougr** utilizando la imagen oficial locustio/locust. Este servicio actuará como nodo maestro en la configuración de Locust, coordinando múltiples workers.
- Se montará un volumen para incluir el archivo locustfile.py, que contiene la definición de las pruebas.
- Este contenedor se unirá a la red_web y se le asignará una dirección IP estática de 192.168.10.70.

Definición del Servicio Worker:

- Se configurará un servicio **worker-tuusuariougr** que también utilizará la imagen locustio/locust. Este servicio funcionará como nodos workers que simulan usuarios bajo la coordinación del master.
- Los workers también estarán en la red_web y se escalarán a 5 réplicas para aumentar la capacidad de simulación.

Ejemplo básico de DockerCompose.yml

```
services:
  master-tuusuariougr:
    image: locustio/locust
    ports:
     - "8089:8089"
    volumes:
      - ./:/mnt/locust
    command: -f /mnt/locust/locustfile.py --master -H https://192.168.10.50:443/
    networks:
      red_web:
        ipv4_address: 192.168.10.70

  worker-tuusuariougr:
    image: locustio/locust
    volumes:
      - ./:/mnt/locust
    command: -f /mnt/locust/locustfile.py --worker --master-host master-tuusuariougr
    depends_on:
      - master-tuusuariougr
    deploy:
      replicas: 5
    networks:
      - red_web


networks:
  red_web:
    external: true
```

2.2 Archivo de Configuración Locust (locustfile.py):

Definición de la Clase de Tareas:
- Se definirá una clase P5_tuusuariougr dentro de locustfile.py que incluirá tareas específicas para hacer peticiones GET a index.php.
- Se deshabilitará la verificación SSL con verify=False para permitir peticiones a servidores con certificados autofirmados.

Configuración de Tiempos de Espera:
- La clase P5_usuarios simulará a los usuarios que realizan las pruebas, con una espera aleatoria entre 1 y 5 segundos entre cada acción para emular el comportamiento humano más realista.

Ejemplo básico de Locustfile.py

```
from locust import HttpUser, TaskSet, task, between

class P5_tuusuariougr(TaskSet):
    @task
    def load_index(self):
        self.client.get("/index.php", verify=False)

class P5_usuarios(HttpUser):
    tasks = [P5_tuusuariougr]
    wait_time = between(1, 5)
```

Parte 3: Verificación y Pruebas del escenario con Benchmarking

En esta sección se centra en la ejecución y evaluación de las pruebas de carga configuradas en las secciones anteriores, utilizando tanto Apache Benchmark (ab) como Locust. El objetivo es validar la eficacia de la granja web bajo carga y analizar cómo las configuraciones de seguridad, balanceo de carga y SSL afectan el rendimiento: al menos 10000 peticiones y 100 usuarios en concurrencia.

3.1 Despliegue y Ejecución:

1. Despliegue de la Granja Web:
Se ejecutará el comando docker-compose up -d para iniciar todos los servicios definidos en el archivo docker-compose.yml. Esto incluye la granja web con SSL y IPTABLES configurados, junto con los servicios de benchmarking de ab y Locust.
Se asegurará que todos los contenedores estén activos y funcionando correctamente.

2. Ejecución de Pruebas con Apache Benchmark:
Desde el contenedor de ab, se lanzarán peticiones HTTP y HTTPS dirigidas al balanceador de carga en la dirección IP 192.168.10.50.
Se evaluará la capacidad de respuesta del sistema ante un volumen alto de peticiones, monitorizando la latencia, el número de peticiones por segundo y la tasa de errores.

3. Ejecución de Pruebas con Locust:
Iniciará el servicio Locust a través de su interfaz web accesible desde el navegador, permitiendo un control más detallado y visual de las pruebas de carga.
Se lanzarán simulaciones de tráfico desde el nodo master Locust, distribuyendo tareas a través de los nodos worker, para simular comportamientos de usuario real en el acceso a la granja web.

3.2 Análisis de los Test de Carga:

En este apartado se analizarán los resultados de los distintos test de carga realizados con **ab** y con **Locust** para ver cómo influye en el **rendimiento** peticiones **http vs https** y configuraciones avanzadas de prácticas anteriores respecto a **optimizaciones**, etc.

1. Análisis de Resultados de Apache Benchmark:

Se recogerán y analizarán los resultados obtenidos de las pruebas de ab, centrándose en identificar cuellos de botella y evaluar la efectividad de las configuraciones de SSL, protocolo TLS, reglas IPTABLES bajo carga.

Se prestará especial atención a las métricas de rendimiento general en **peticiones por segundo**, **tiempo de respuesta, conexiones, tiempo de espera** y **errores** para determinar si las configuraciones actuales sostienen un rendimiento óptimo.

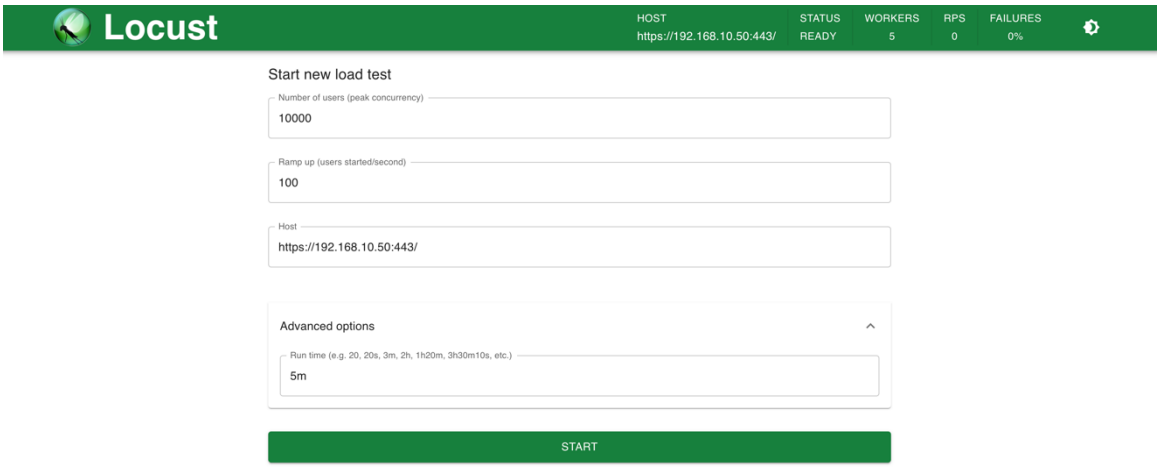
Ejemplo:

```
apache_benchmark-P5  | This is ApacheBench, Version 2.3 <$Revision: 1913912 $>
apache_benchmark-P5  | Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
apache_benchmark-P5  | Licensed to The Apache Software Foundation, http://www.apache.org/
apache_benchmark-P5  |
apache_benchmark-P5  | Benchmarking 192.168.10.50 (be patient)
apache_benchmark-P5  | Completed 1000 requests
apache_benchmark-P5  | Completed 2000 requests
apache_benchmark-P5  | Completed 3000 requests
apache_benchmark-P5  | Completed 4000 requests
apache_benchmark-P5  | Completed 5000 requests
apache_benchmark-P5  | Completed 6000 requests
apache_benchmark-P5  | Completed 7000 requests
apache_benchmark-P5  | Completed 8000 requests
apache_benchmark-P5  | Completed 9000 requests
apache_benchmark-P5  | Completed 10000 requests
apache_benchmark-P5  | Finished 10000 requests
apache_benchmark-P5  |
apache_benchmark-P5  |
apache_benchmark-P5  | Server Software:        nginx/1.25.5
apache_benchmark-P5  | Server Hostname:        192.168.10.50
apache_benchmark-P5  | Server Port:            443
apache_benchmark-P5  | SSL/TLS Protocol:       TLSv1.3,TLS_AES_256_GCM_SHA384,2048,256
apache_benchmark-P5  | Server Temp Key:        X25519 253 bits
apache_benchmark-P5  |
apache_benchmark-P5  | Document Path:          /
apache_benchmark-P5  | Document Length:        332 bytes
apache_benchmark-P5  |
apache_benchmark-P5  | Concurrency Level:      100
apache_benchmark-P5  | Time taken for tests:   9.313 seconds
apache_benchmark-P5  | Complete requests:      10000
apache_benchmark-P5  | Failed requests:        0
apache_benchmark-P5  | Total transferred:      5130000 bytes
apache_benchmark-P5  | HTML transferred:       3320000 bytes
apache_benchmark-P5  | Requests per second:    1073.71 [#/sec] (mean)
apache_benchmark-P5  | Time per request:       93.135 [ms] (mean)
apache_benchmark-P5  | Time per request:       0.931 [ms] (mean, across all concurrent requests)
apache_benchmark-P5  | Transfer rate:          537.90 [Kbytes/sec] received
apache_benchmark-P5  |
apache_benchmark-P5  | Connection Times (ms)
apache_benchmark-P5  |               min  mean[+/-sd] median   max
apache_benchmark-P5  | Connect:        1    2   2.1      1      62
apache_benchmark-P5  | Processing:     5   91  57.8     75     538
apache_benchmark-P5  | Waiting:        3   91  57.8     75     538
apache_benchmark-P5  | Total:          6   93  57.8     77     539
apache_benchmark-P5  |
apache_benchmark-P5  | Percentage of the requests served within a certain time (ms)
apache_benchmark-P5  |   50%     77
apache_benchmark-P5  |   66%     98
apache_benchmark-P5  |   75%    114
apache_benchmark-P5  |   80%    128
apache_benchmark-P5  |   90%    166
apache_benchmark-P5  |   95%    204
apache_benchmark-P5  |   98%    262
apache_benchmark-P5  |   99%    303
apache_benchmark-P5  |  100%    539 (longest request)
```

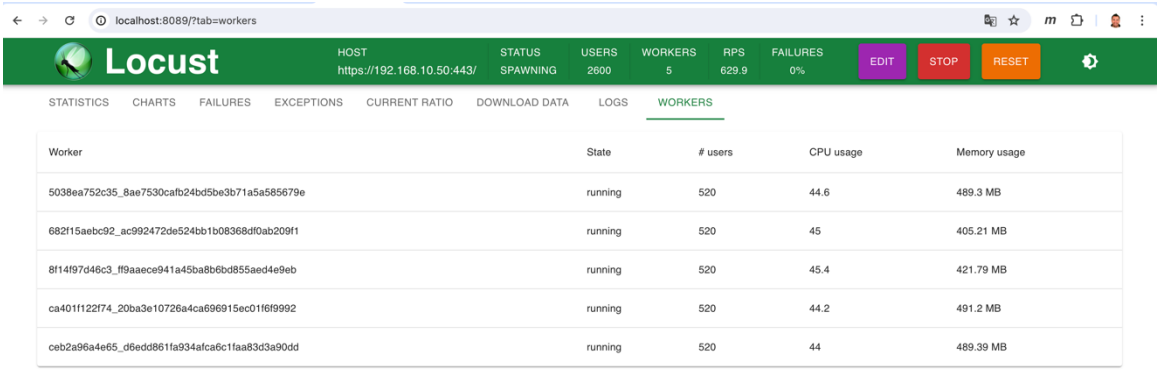
2. Análisis de Resultados de Locust:
Se analizarán los datos recogidos por Locust, incluyendo el número de usuarios simulados, las tasas de fallo, tiempos de respuesta y la distribución de carga entre los servidores.

Se prestará especial atención a las métricas de rendimiento general en **total de solicitudes, solicitudes fallidas, tiempos de respuesta, conexiones, tiempo de espera** y **errores** para determinar si las configuraciones actuales sostienen un rendimiento óptimo.

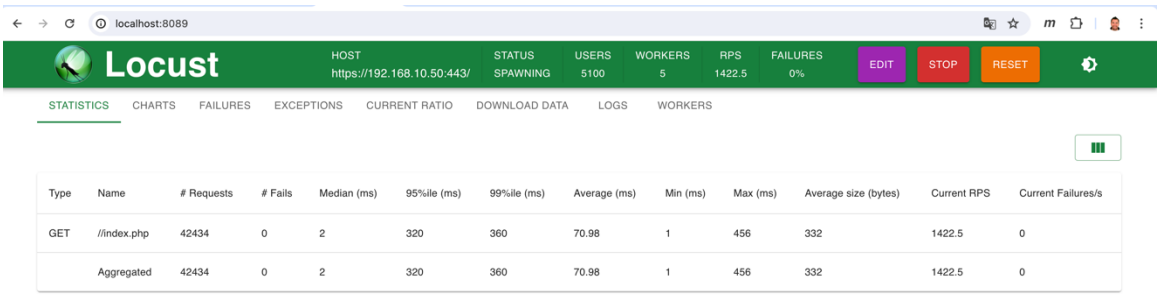
Ejemplo test de carga



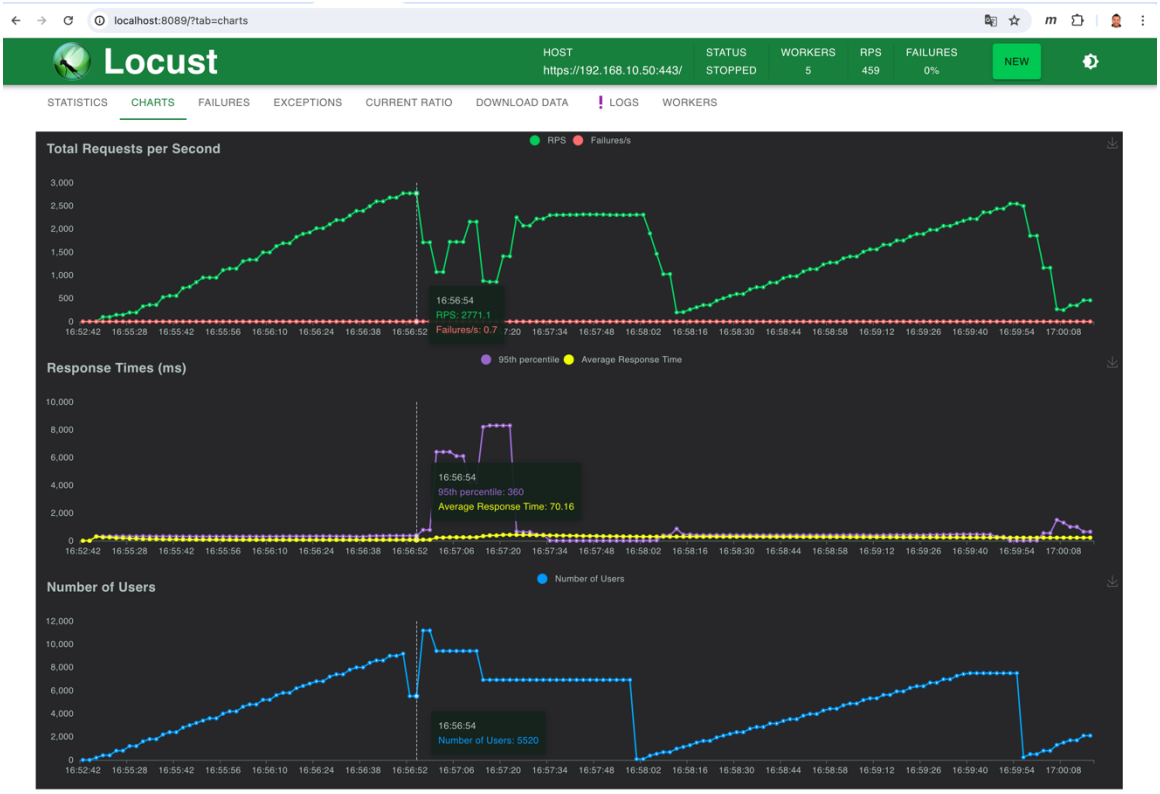
Ejemplo de los workers

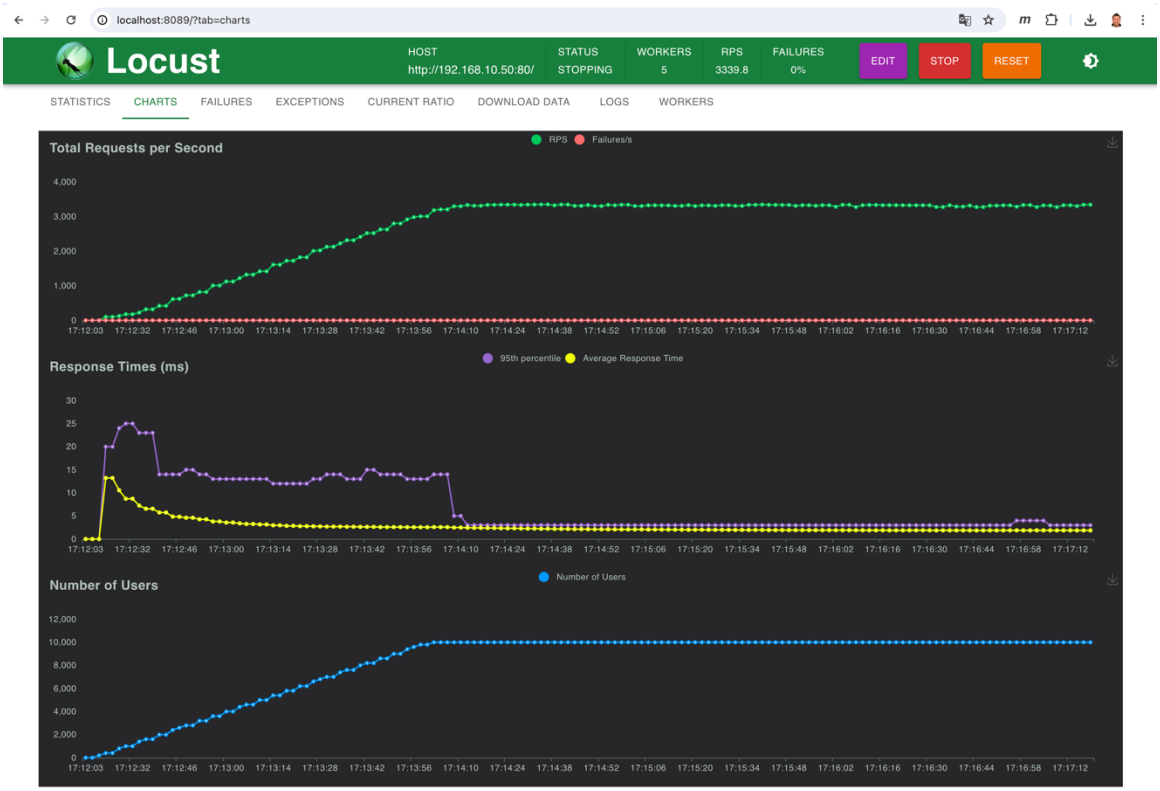


Ejemplo resumen de métricas



Ejemplo de gráficos de test de carga





Evaluación

La práctica se realizará de manera individual. Tiene un peso del **15%** del total de prácticas.

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

B1: **Configuración del entorno de benchmarking**
- Preparación de un entorno de trabajo específico para realizar tests de carga.
- Creación de directorios y archivos necesarios para los tests con Apache Benchmark (AB) y Locust.

B2: **Implementación con Apache Benchmark**
- Desarrollo de un Dockerfile (DockerFileAB) que configure un contenedor para ejecutar AB.
- Ejecución de AB desde un contenedor, lanzando peticiones HTTP y HTTPS al balanceador en la dirección 192.168.10.50.
- Asegurar que el contenedor AB esté en la misma red (red_web) que el balanceador.

B3: **Implementación con Locust**
- Configuración de un docker-compose.yml para ejecutar Locust con un nodo master y múltiples workers (ejemplo: 5).
- Preparación del archivo locustfile.py con tareas definidas para realizar peticiones HTTP y HTTPS a la granja web y tiempo aleatorio entre peticiones.
- Establecimiento de la configuración necesaria para permitir peticiones HTTPS a un sitio con certificado autofirmado (uso de verify=False).

B4: **Ejecución de pruebas de carga**
- Despliegue y ejecución de los escenarios con Apache Benchmark y Locust, monitorizando el comportamiento y el rendimiento de la granja web bajo carga.
- Pruebas realizadas con al menos 10000 peticiones y 100 usuarios en concurrencia así como peticiones https y http.

B5: **Análisis de resultados**
- Revisión y análisis de los datos obtenidos de las pruebas de carga ejecutadas con AB y Locust.
- Documentación de los resultados, incluyendo métricas como solicitudes por segundo, tiempos de respuesta y errores.

Se proponen, opcionalmente, las siguientes **tareas avanzadas:**

A1: Desarrollar tareas avanzadas en Locustfile.py
- Definir tareas que incluyan la navegación por páginas, la creación o interacción con contenido, y las interacciones con la base de datos como insertar comentarios o publicaciones.
- Incorporar tareas que reflejen operaciones típicas del CMS, como la autenticación de usuarios, la carga de múltiples tipos de contenido, y la ejecución de consultas de búsqueda.

A2. Crear escenario multicontenedor con algún CMS

- Configurar contenedores Docker adicionales para cada instancia de un CMS seleccionado, como Drupal, WordPress, o Moodle, integrados con la granja web existente.
- Asegurar la configuración adecuada del balanceador de carga para dirigir el tráfico hacia las instancias del CMS.
- Establecer conexiones a bases de datos adecuadas y asegurarse de que todas las instancias del CMS puedan realizar operaciones de lectura y escritura de manera eficiente.
- Conectar base de datos a red_servicios de la granja.

A3. Ejecución y Análisis de cargas de prueba avanzadas sobre CMS

- Desplegar el escenario con el balanceador de carga y las instancias del CMS configuradas para las pruebas.
- Lanzar pruebas de carga desde el contenedor master de Locust y coordinar a los workers para generar tráfico significativo hacia el CMS, evaluando la capacidad del sistema para manejar varias solicitudes simultáneas y operaciones de base de datos.
- Analizar los resultados proporcionados por Locust, incluyendo el número de usuarios simultáneos que el sistema puede soportar, los tiempos de respuesta y la tasa de errores.
- Determinar los cuellos de botella y las limitaciones de rendimiento, y proporcionar recomendaciones para optimizar la configuración del CMS y la infraestructura de la granja web.

Normas de entrega

Se entregará un documento .pdf con el desarrollo de la práctica según el guion **detallando** e indicando, en su caso, los **aspectos básicos y avanzados realizados**, comandos de terminal ejecutados, así como las configuraciones o soluciones proporcionadas por la IA generativa y las configuraciones o soluciones que finalmente utiliza el estudiante junto con su análisis crítico. 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 - B2: Implementación con Apache Benchmark* donde aparezcan detalladas las configuraciones, explicaciones sobre ellas y resultados proporcionados por la IA generativa y **un análisis de éstos**. De igual forma, si por ejemplo, se han realizado tareas avanzadas sobre automatizaciones con Scripts, debe aparecer *Tareas Avanzadas – A1: Desarrollar tareas avanzadas en Locustfile.py,* detalles de las configuraciones, explicaciones sobre ellas y resultados proporcionados por la IA generativa y **un análisis de éstos**.

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.

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 ApellidosNombreP5.pdf
2. Un archivo .zip con los distintos archivos de configuraciones, carpetas, etc. necesarios para la ejecución de la práctica siguiendo el formato ApellidosNombreP5.zip

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.

Las faltas de ortografía en la redacción se penalizarán con hasta 2 puntos de la nota de la práctica.

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 Configuración del entorno de benchmarking (5 puntos)	Entorno correctamente configurado con todos los directorios y archivos necesarios, perfectamente documentado. **(5 puntos)**	Entorno bien configurado con mínimos errores documentales o en la estructura de archivos. **(3.75 puntos)**	Configuración incompleta o con errores significativos. **(2.5 puntos)**	Configuración incorrecta que afecta la ejecución de pruebas. **(1.25 puntos)**	No se realizó la configuración del entorno. **(0 puntos)**
B2: Implementación con Apache Benchmark (20 puntos)	Dockerfile y ejecución de AB perfectamente configurados, documentados y ejecutados sin errores. **(20 puntos)**	Configuración funcional con algunos errores no críticos. **(15 puntos)**	Configuraciones con errores que requieren ajustes menores. **(10 puntos)**	Errores significativos que afectan la ejecución de AB. **(5 puntos)**	No realizado. **(0 puntos)**
B3: Implementación con Locust (30 puntos)	Configuración correcta y ejecución de Locust con todas las especificaciones cumplidas. **(30 puntos)**	Implementación adecuada con pequeños errores o falta de detalle. **(22.5 puntos)**	Errores en la configuración que afectan parcialmente las pruebas. **(15 puntos)**	Configuración incorrecta que afecta significativamente las pruebas. **(7.5 puntos)**	No realizado. **(0 puntos)**
B4: Ejecución de pruebas de carga (10 puntos)	Ejecución correcta con más de 10000 peticiones y 100 usuarios concurrentes, incluyendo HTTP y HTTPS, con documentación detallada. **(10 puntos)**	Ejecución con pequeños errores, pero alcanza la mayoría de los requisitos de peticiones y concurrencia. **(7.5 puntos)**	Errores notables que afectan algunas métricas de rendimiento, no cumple todos los requisitos. **(5 puntos)**	Múltiples errores, rendimiento bajo carga no evaluado adecuadamente. **(2.5 puntos)**	No realizado. **(0 puntos)**
B5: Análisis de resultados (30 puntos)	Análisis exhaustivo y detallado de todos los resultados con métricas claras. **(30 puntos)**	Análisis adecuado con algunos detalles menores omitidos. **(22.5 puntos)**	Análisis superficial con omisiones significativas. **(15 puntos)**	Análisis inadecuado con información confusa o incorrecta. **(7.5 puntos)**	No realizado. **(0 puntos)**
Tareas avanzadas					
A1: Desarrollo avanzado en Locustfile.py (20 puntos)	Tareas bien definidas que simulan interacciones complejas con el CMS y pruebas de funcionalidades específicas. **(20 puntos)**	Ejecución con inexactitudes menores, cubre la mayoría de funcionalidades pero con algunas omisiones. **(15 puntos)**	Tareas básicas realizadas sin ajustes avanzados. **(10 puntos)**	Ejecución con errores significativos, no cubre funcionalidades clave del CMS. **(5 puntos)**	No realizado. **(0 puntos)**
A2: Creación de escenario multicontenedor con CMS (20 puntos)	Configuración perfecta de contenedores para CMS con integración eficiente y rendimiento óptimo. **(20 puntos)**	Configuración correcta con algunos errores menores, integración adecuada del CMS. **(15 puntos)**	Configuraciones básicas completadas, algunas deficiencias en integración o rendimiento. **(10 puntos)**	Errores graves que afectan la funcionalidad de los CMS. **(5 puntos)**	No realizado. **(0 puntos)**

A3: Ejecución y análisis de cargas de prueba avanzadas sobre CMS (20 puntos)	Ejecución meticulosa con análisis detallado de capacidad, respuesta y tasa de errores, identificando claramente los cuellos de botella. **(20 puntos)**	Pruebas bien realizadas con análisis adecuado, algunas áreas sin explorar completamente. **(15 puntos)**	Ejecución de pruebas con análisis básico, falta de detalle en identificación de problemas. **(10 puntos)**	Pruebas ejecutadas con múltiples errores, análisis insuficiente de los resultados. **(5 puntos)**	No realizado. **(0 puntos)**

Documentación					
Documentación de la Práctica (25 puntos)	Memoria detallada, cuidada, bien estructurada, sin errores ortográficos. **(25 puntos)**	Memoria completa con algunos detalles menores faltantes o errores leves. **(18.75 puntos)**	Memoria completa pero con varias omisiones o errores ortográficos. **(12.5 puntos)**	Memoria incompleta, desorganizada o con numerosos errores. **(6,25 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 (20 puntos)	Comentarios detallados y claros en todas las configuraciones. **(20 puntos)**	Comentarios adecuados en la mayoría de las configuraciones. **(15 puntos)**	Algunos comentarios útiles, pero falta de detalle o claridad. **(10 puntos)**	Comentarios escasos o poco claros. **(5 puntos)**	Sin comentarios en las configuraciones. **(0 puntos)**
Penalizaciones					
Requisitos de entrega	Cumple: 0 puntos		No cumple: -15 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