

Servidores Web de Altas Prestaciones



Práctica 5: Benchmarking



ICAR

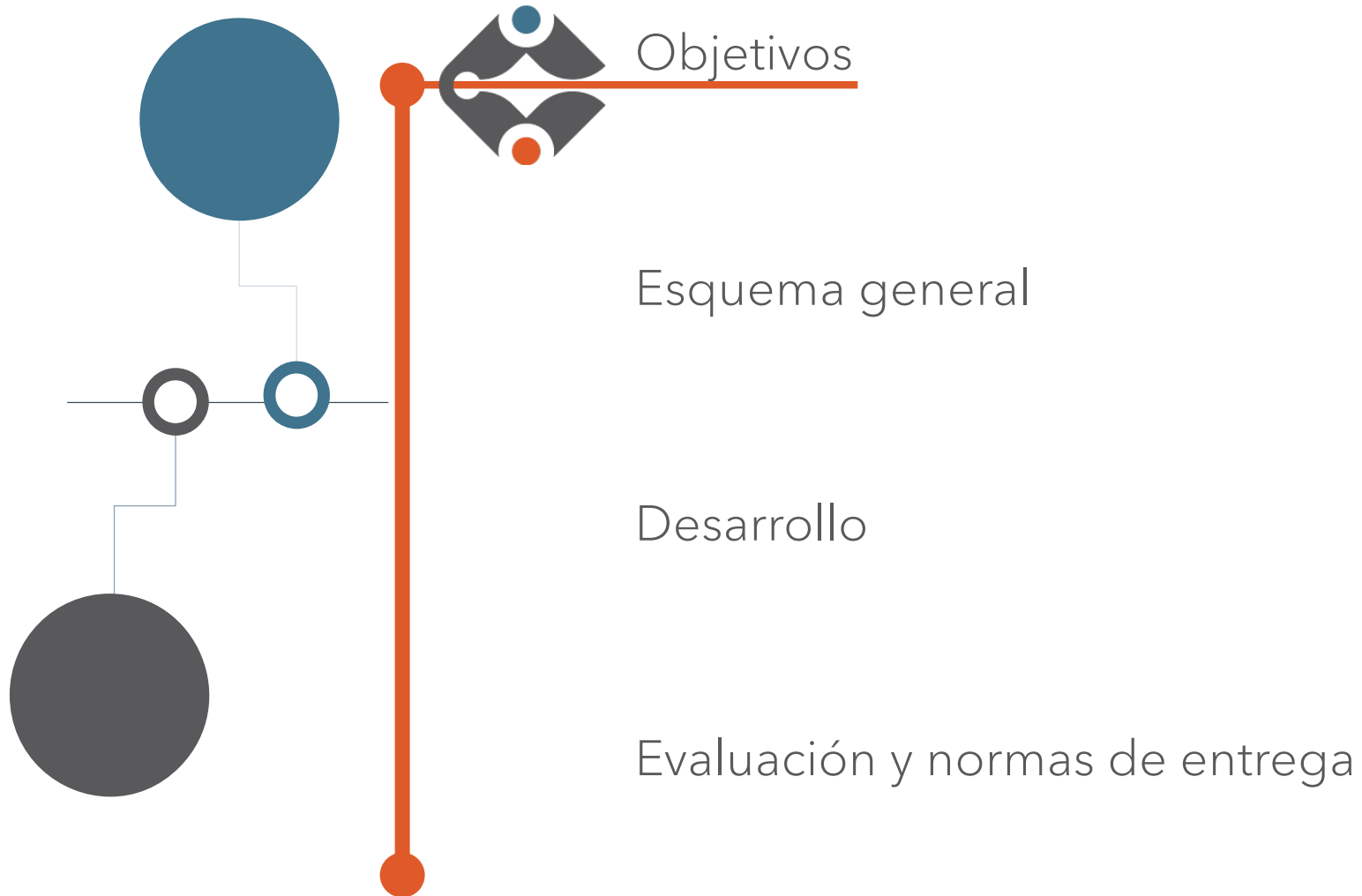
INGENIERÍA DE COMPUTADORES,
AUTOMÁTICA Y ROBÓTICA



UNIVERSIDAD
DE GRANADA



Índice





Objetivos

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.

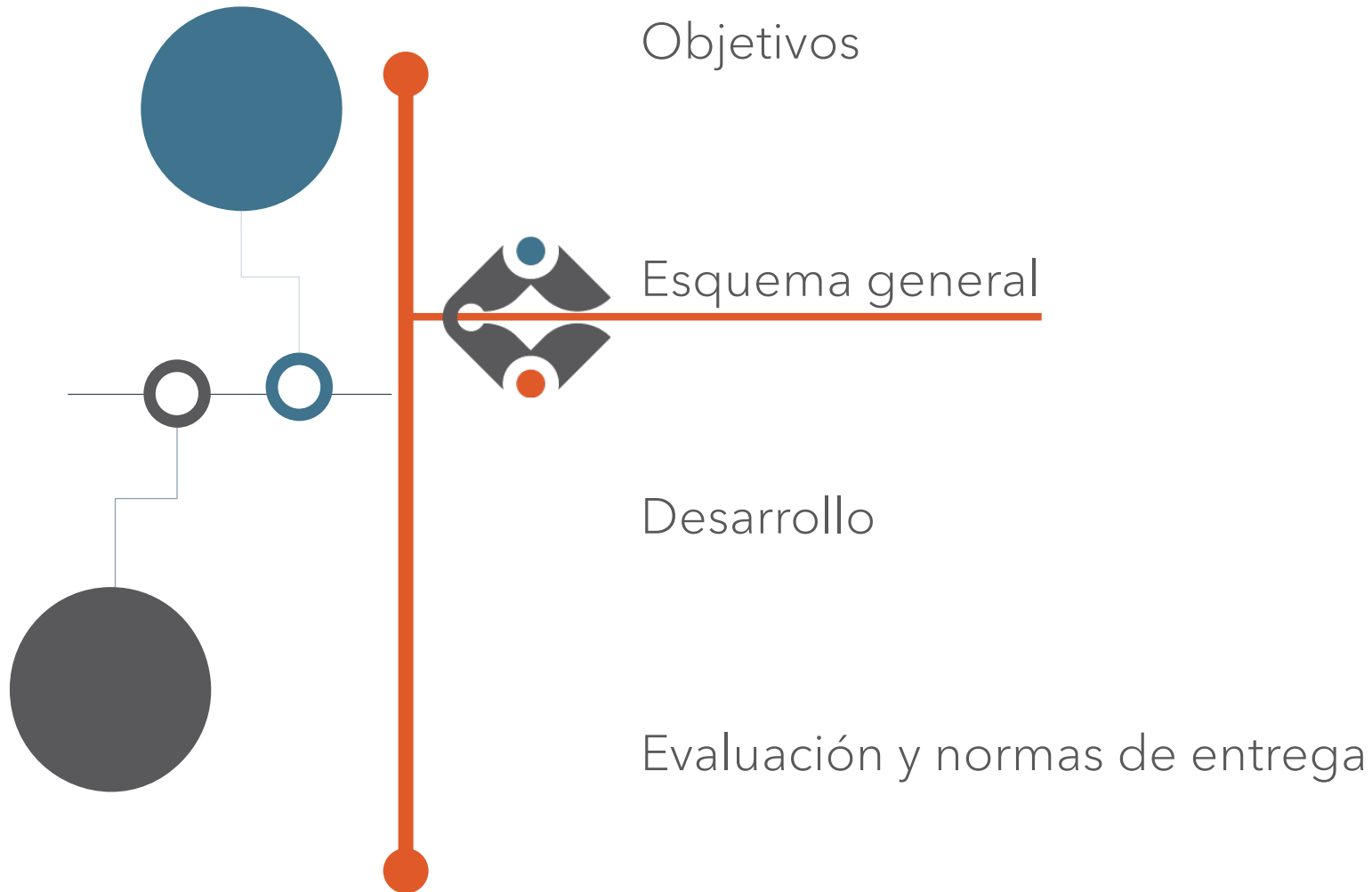
1. Configurar y utilizar Apache Benchmark para realizar pruebas básicas de carga en la granja web.
2. Implementar Locust en un escenario de múltiples contenedores para simular tráfico de usuarios y evaluar el rendimiento del sistema bajo carga.
3. Analizar y comparar los resultados de las pruebas para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

La práctica se realizará de manera individual. Tiene un peso del **20%** del total de prácticas.





Índice





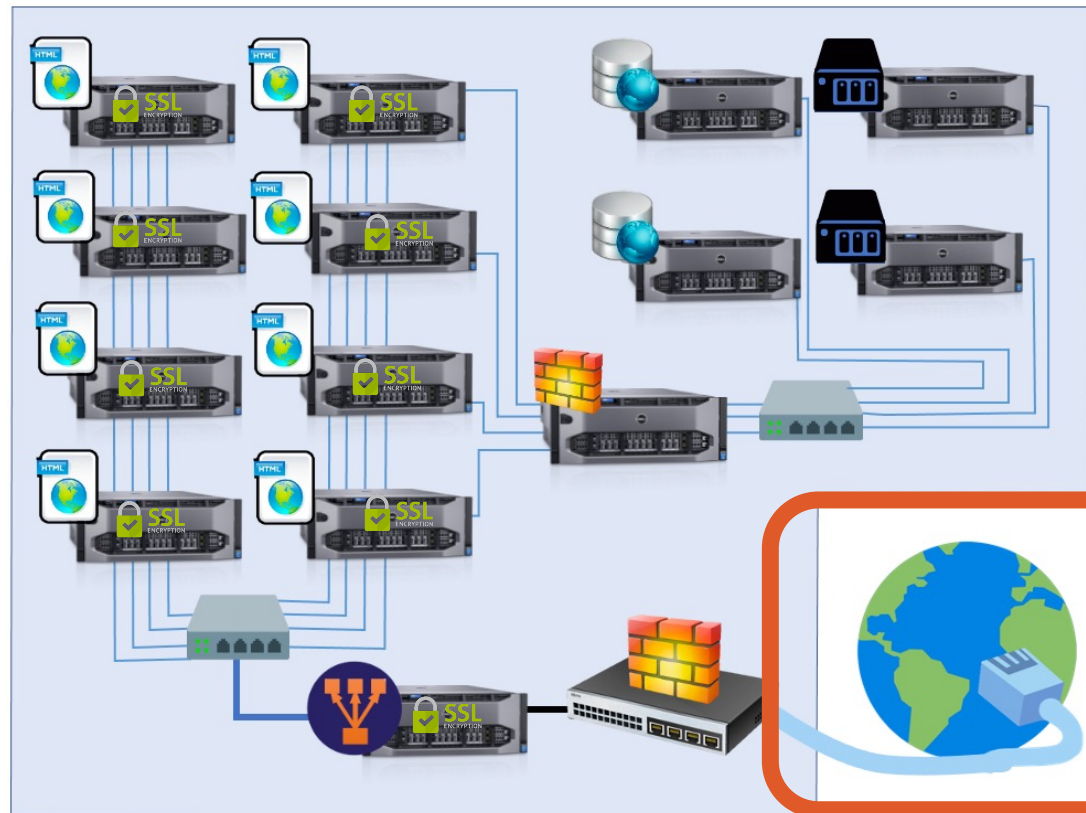
Esquema general



- Test de carga básico (ab)
- Test de carga personalizable (Locust)
- Análisis del rendimiento



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.

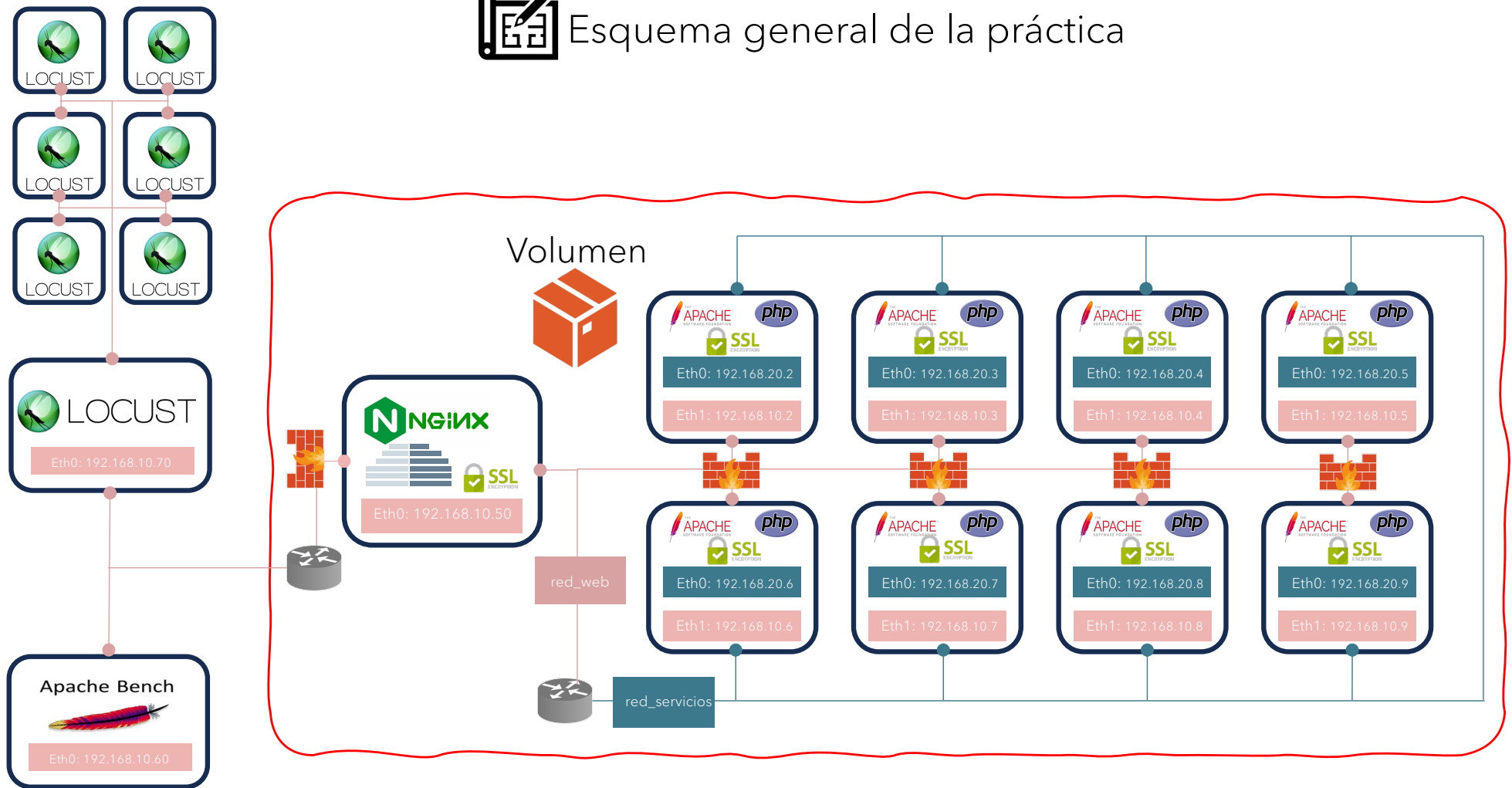




Esquema general



Esquema general de la práctica





Índice





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 parte se configurará y utilizará Apache Benchmark (ab) para lanzar pruebas de rendimiento contra el balanceador de carga de la granja web. Trabajaremos en el directorio P5-ab.

Dockerfile para Apache Benchmark (DockerFileAB):

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
```





Parte 1: Implementación de Apache Benchmark

En esta parte se configurará y utilizará Apache Benchmark (ab) para lanzar pruebas de rendimiento contra el balanceador de carga de la granja web. Trabajaremos en el directorio P5-ab.

Integración con Docker Compose:

- Definición del Servicio:

En el docker-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.





Parte 1: Implementación de Apache Benchmark

En esta parte se configurará y utilizará Apache Benchmark (ab) para lanzar pruebas de rendimiento contra el balanceador de carga de la granja web. Trabajaremos en el directorio **P5-ab**.

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 con Locust

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**.

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 6 réplicas para aumentar la capacidad de simulación.





Parte 2: Implementación con Locust

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**.

Docker Compose para Locust

Ejemplo de docker-compose.yml (1/2) - master

```
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
```





Parte 2: Implementación con Locust

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**.

Docker Compose para Locust

Ejemplo de docker-compose.yml (2/2) - worker

```
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: 6
```





Parte 2: Implementación con Locust

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**.

Archivo 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.





Parte 2: Implementación con Locust

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**.

Archivo configuración Locust - locustfile.py

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. Al menos 10000 peticiones y 100 usuarios en concurrencia.

Despliegue y Ejecución:

1. **Despliegue de la Granja Web:** se ejecutará el comando `docker-compose up -d`
2. **Ejecución de Pruebas con Apache Benchmark:** desde el contenedor de ab, se lanzarán peticiones HTTP y HTTPS dirigidas al balanceador de carga (192.168.10.50).
3. **Ejecución de Pruebas con Locust:** 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.





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. Al menos 10000 peticiones y 100 usuarios en concurrencia.

Análisis de Resultados de Apache Benchmark:

Métricas de rendimiento general en peticiones por segundo, tiempo de respuesta, conexiones, tiempo de espera y errores.

```
apache_benchmark-P5 | This is ApacheBench, Version 2.3 <$Revision: 1913912 $>
apache_benchmark-P5 | Copyright 1996 Adam Twiss, Zeus Technology Ltd, ...
apache_benchmark-P5 | Licensed to The Apache Software Foundation, ...
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 | 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 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)
```





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. Al menos 10000 peticiones y 100 usuarios en concurrencia.

Análisis de Resultados de Locust:

Métricas de rendimiento general en total de solicitudes, solicitudes fallidas, tiempos de respuesta, conexiones, tiempo de espera y errores.

Ejemplo test de
carga

The screenshot shows the Locust web interface. At the top, there's a green header with the Locust logo and a status bar. The status bar includes fields for HOST (https://192.168.10.50:443/), STATUS (READY), WORKERS (5), RPS (0), and FAILURES (0%). Below the header, there's a section titled "Start new load test". It contains four input fields: "Number of users (peak concurrency)" with the value 10000, "Ramp up (users started/second)" with the value 100, "Host" with the value https://192.168.10.50:443/, and "Advanced options" which is expanded to show "Run time (e.g. 20s, 3m, 2h, 1h20m, 3h30m10s, etc.)" with the value 5m. At the bottom of this section is a large green button labeled "START".





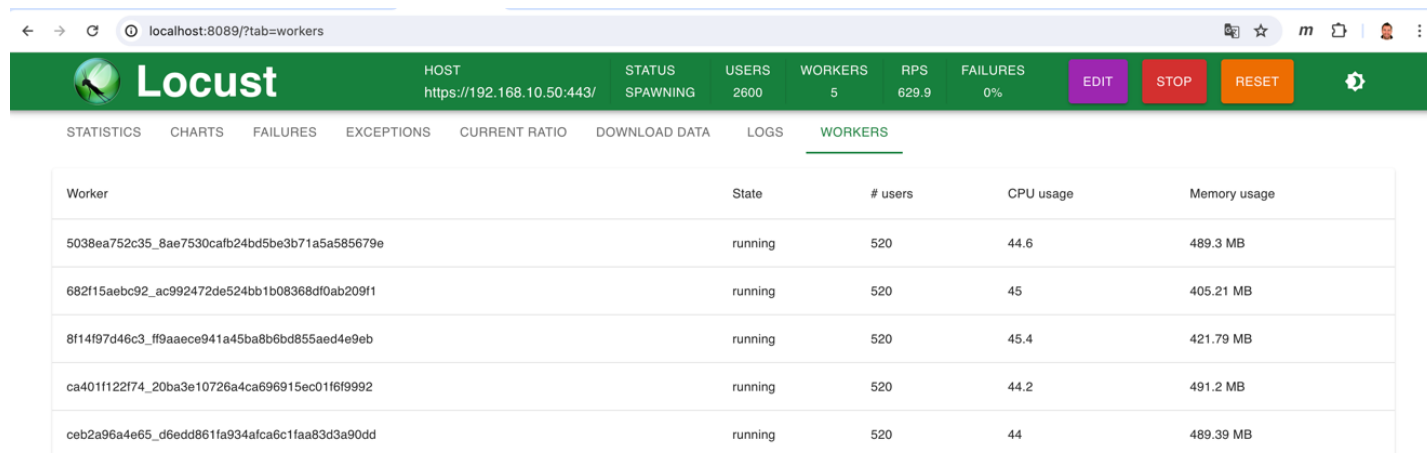
Parte 3: Verificación y Pruebas del escenario con Benchmarking

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. Al menos 10000 peticiones y 100 usuarios en concurrencia.

Análisis de Resultados de Locust:

Métricas de rendimiento general en total de solicitudes, solicitudes fallidas, tiempos de respuesta, conexiones, tiempo de espera y errores.

Ejemplo de los
workers



Worker	State	# users	CPU usage	Memory usage
5038ea752c35_8ae7530cafb24bd5be3b71a5a585679e	running	520	44.6	489.3 MB
682f15aebc92_ac992472de524bb1b08368df0ab209f1	running	520	45	405.21 MB
8f14f97d46c3_f9aaece941a45ba8b6bd855aed4e9eb	running	520	45.4	421.79 MB
ca401f122f74_20ba3e10726a4ca696915ec01f6f9992	running	520	44.2	491.2 MB
ceb2a96a4e65_d6edd861fa934afca6c1faa83d3a90dd	running	520	44	489.39 MB





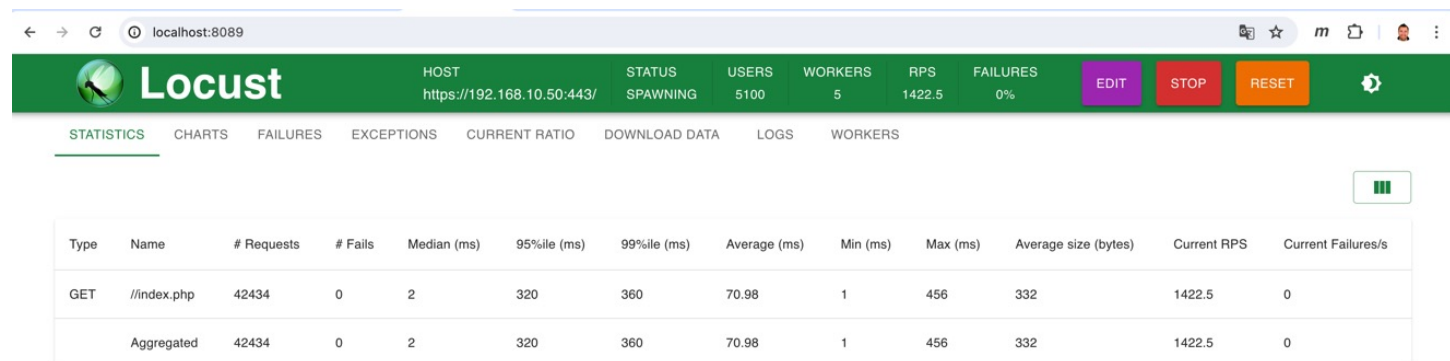
Parte 3: Verificación y Pruebas del escenario con Benchmarking

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. Al menos 10000 peticiones y 100 usuarios en concurrencia.

Análisis de Resultados de Locust:

Métricas de rendimiento general en total de solicitudes, solicitudes fallidas, tiempos de respuesta, conexiones, tiempo de espera y errores.

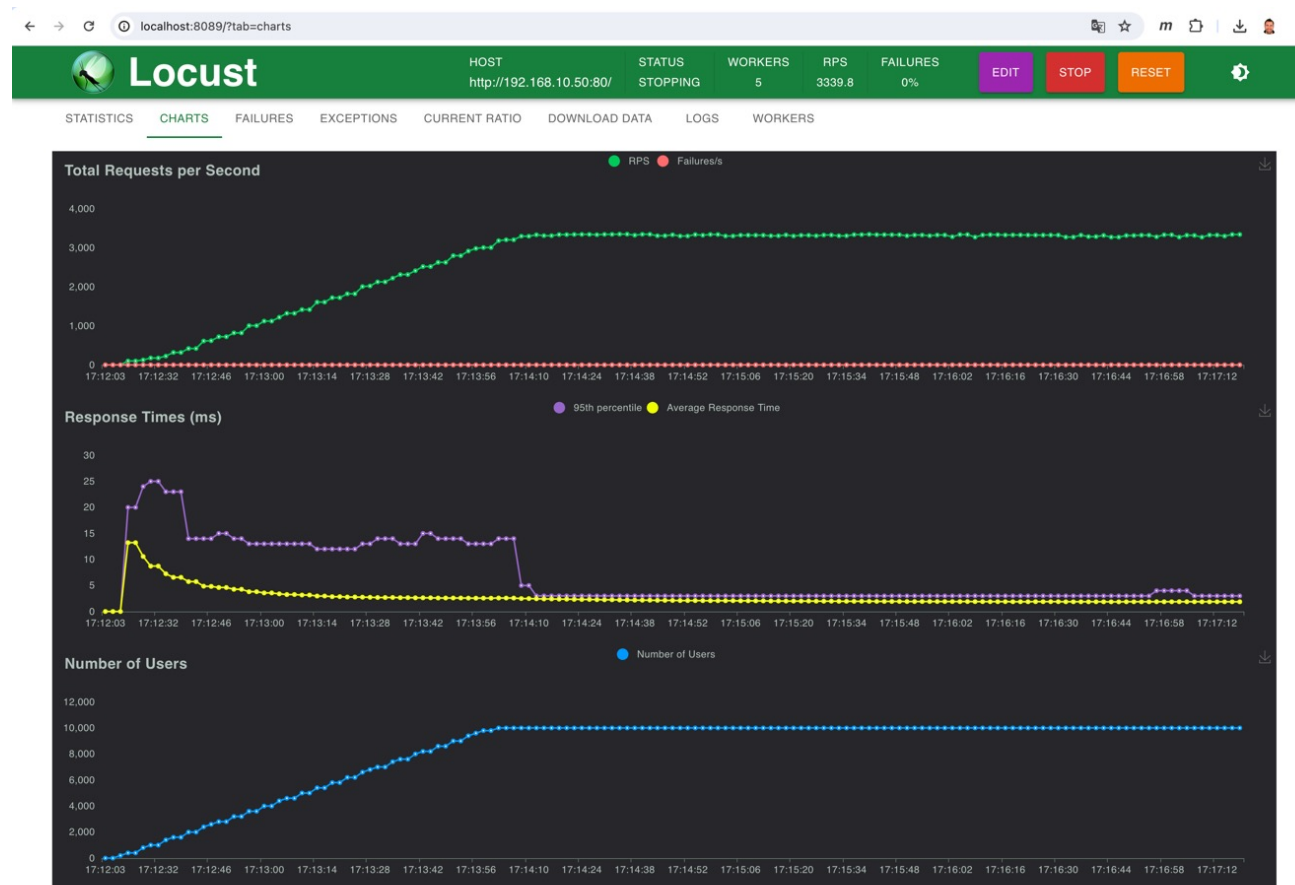
Ejemplo resumen de métricas





Parte 3: Verificación y Pruebas del escenario con Benchmarking

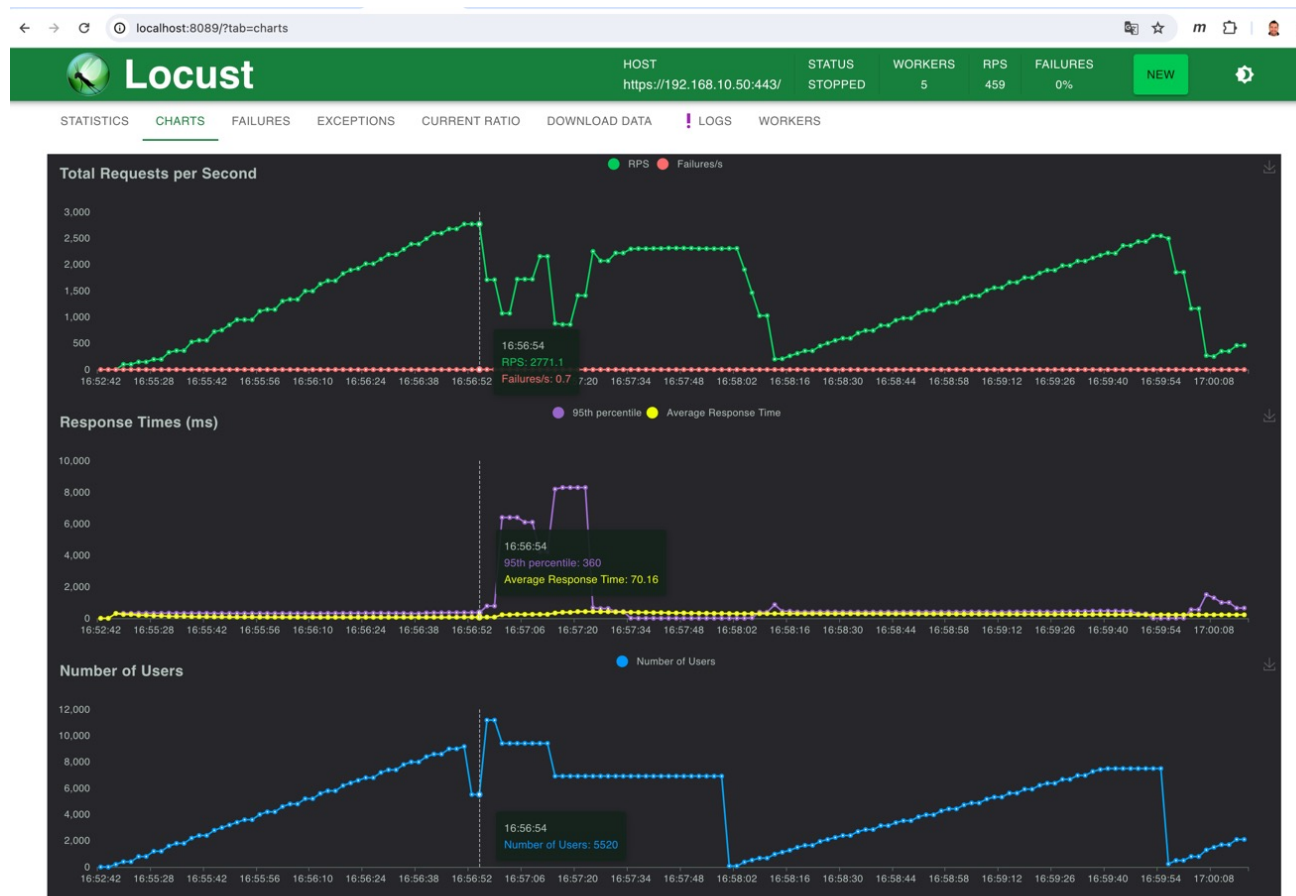
Análisis de Resultados de Locust - ejemplo gráfico de test de carga





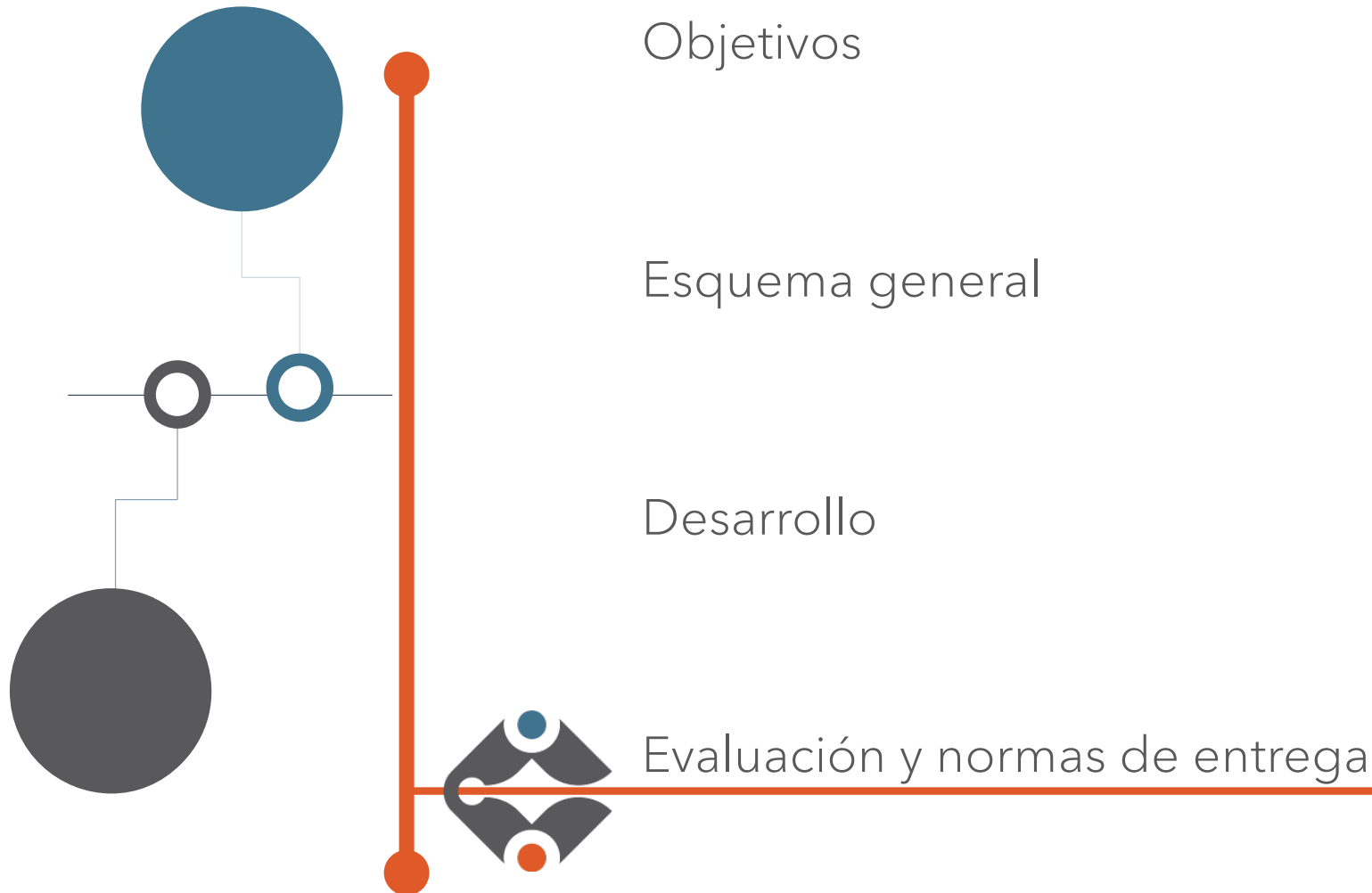
Parte 3: Verificación y Pruebas del escenario con Benchmarking

Análisis de Resultados de Locust - ejemplo gráfico de test de carga





Índice





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.





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



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.





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

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.



Se proponen, opcionalmente, las siguientes tareas avanzadas:



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.



Se proponen, opcionalmente, las siguientes tareas avanzadas:



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.





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 - Tareas Básicas - B2. Implementación con Apache Benchmark* donde aparezcan detalladas las configuraciones y resultados 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, resultados y un análisis de éstos.

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 **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**

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.





La práctica se realizará de manera individual. Tiene un peso del **20%** del total de prácticas.

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.

Cuestiones sobre la calificación obtenida en cada práctica se realizarán **UNICAMENTE** en la sesión dedicada a recuperación/defensa al final de curso.

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.



Servidores Web de Altas Prestaciones



Práctica 5: Benchmarking



ICAR

INGENIERÍA DE COMPUTADORES,
AUTOMÁTICA Y ROBÓTICA



UNIVERSIDAD
DE GRANADA