

P2 Balanceo de Carga

Mario Rodriguez Lopez

June 2024

Índice

B1.CONFIGURACIÓN DEL ENTORNO DE BENCHMARKING	2
B2.IMPLEMENTACIÓN CON APACHE BENCHMARK	2
B3.IMPLEMENTACIÓN CON LOCUST	2
B4.EJECUCIÓN DE PRUEBAS DE CARGA	3
B5.ANALISIS DE RESULTADOS	11
A1.DESARROLLAR TAREAS AVANZADAS EN LOCUSTFILE.PY	12
A2.CREAR ESCENARIO MULTICONTENEDOR CON ALGÚN CMS	15
A3.EJECUCIÓN Y ANÁLISIS DE CARGAS DE PRUEBA AVANZADAS SOBRE CMS	23

B1.CONFIGURACIÓN DEL ENTORNO DE BENCHMARKING

Para crear los directorios simplemente uso lo siguiente:

```
mkdir P5-granjaweb
mkdir P5-ab
mkdir P5-locust
```

Después copio todo lo de la P4 en P5-granjaweb

No he usado IA para realizar esto puesto que es muy simple.

B2.IMPLEMENTACIÓN CON APACHE BENCHMARK

Siguiendo el guión de prácticas comienzo creando el Dockerfile:

```
FROM debian:latest

RUN apt-get update && apt-get install -y apache2-utils
```

Después creo el fichero docker compose con el que lanzaré mi servicio de apache benchmark una vez que haya lanzado todos los otros contenedores. El fichero siguiendo el guión me queda como sigue:

```
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:
      p5-granjaweb_red_web:
        ipv4_address: 192.168.10.60
networks:
  p5-granjaweb_red_web:
    external: true
```

Para que se conecte a la misma red debemos ver el nombre de la red una vez que lanzamos el docker compose de la granja web y ese será el nombre que debemos usar, en este caso p5-granjaweb_red_web. Además el docker compose anterior lanza peticiones https, si queremos mandar peticiones http basta con modificar la dirección a la siguiente:

`http://192.168.10.50:80/`

Y con esto ya podríamos lanzar el test de apache. No he usado IA para realizar esto puesto que he seguido el guión.

B3.IMPLEMENTACIÓN CON LOCUST

Siguiendo el guión de prácticas me queda el siguiente docker compose:

```
services:
  master-mariorgzlpz:
    image: locustio/locust
    ports:
      - "8089:8089"
    volumes:
      - ./:/mnt/locust
    command: -f /mnt/locust/locustfile.py --master -H https://192.168.10.50:443/
    networks:
      p5-granjaweb_red_web:
        ipv4_address: 192.168.10.70

  worker-mariorgzlpz:
    image: locustio/locust
    volumes:
      - ./:/mnt/locust
    command: -f /mnt/locust/locustfile.py --worker --master-host master-mariorgzlpz
```

```

depends_on:
  - master-mariorgzlpz
deploy:
  replicas: 5
networks:
  p5-granjaweb_red_web:

networks:
  p5-granjaweb_red_web:
    external: true

```

Igual que antes el docker compose anterior lanza peticiones https, si queremos mandar peticiones http basta con modificar la dirección a la siguiente:

```
http://192.168.10.50:80/
```

B4.EJECUCIÓN DE PRUEBAS DE CARGA

B4.1 EJECUCIÓN DE APACHE BENCHMARK

Para probar apache benchmark lo unico que debemos hacer es lanzar el servicio con el comando que queramos para hacer las peticiones(http o https) y ver los resultados. Cuando realizo esto aparece el siguiente error:

```

parrot: ~/.../P5/P5-ab
└─ docker compose up
  [+] Running 1/0
    ✓ Container apache_benchmark-P5 Created 0.1s
Attaching to apache_benchmark-P5
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 | Benchmarking 192.168.10.50 (be patient)
apache_benchmark-P5 | SSL read failed (5) - closing connection
apache_benchmark-P5 | SSL read failed (5) - closing connection
apache_benchmark-P5 | SSL read failed (5) - closing connection
apache_benchmark-P5 | SSL read failed (5) - closing connection
apache_benchmark-P5 | SSL read failed (5) - closing connection
apache_benchmark-P5 | SSL read failed (5) - closing connection
apache_benchmark-P5 | SSL read failed (5) - closing connection

```

Para arreglar esto he preguntado a ChatGPT y he modificado el DockerFileAB para añadir nuestro certificado autofirmado a los certificados validos de apache. Me queda lo siguiente:

```

FROM debian:latest

RUN apt-get update && apt-get install -y apache2-utils ca-certificates

COPY ./P5-granjaweb/P4-MarioRgzLpz-certificados/certificado_MarioRgzLpz.crt /usr/local/share/ca-certificates/

RUN update-ca-certificates

```

Después me funcionaba pero realizaba las solicitudes muy lento y esto era porque tenia activada las configuraciones de seguridad avanzadas de la practica 4, por lo que he quitado las configuraciones de seguridad de nginx y de iptables y ya me funcionaba correctamente. Ahora muestro la salida tanto de http como de https:

```

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/1.25.4
apache_benchmark-P5 | Server Hostname:      192.168.10.50
apache_benchmark-P5 | Server Port:          80
apache_benchmark-P5 |
apache_benchmark-P5 | Document Path:        /
apache_benchmark-P5 | Document Length:      88 bytes
apache_benchmark-P5 |
apache_benchmark-P5 | Concurrency Level:    100
apache_benchmark-P5 | Time taken for tests:  1.348 seconds
apache_benchmark-P5 | Complete requests:    10000
apache_benchmark-P5 | Failed requests:      0
apache_benchmark-P5 | Total transferred:    2710000 bytes
apache_benchmark-P5 | HTML transferred:    880000 bytes
apache_benchmark-P5 | Requests per second:  7418.82 [#/sec] (mean)
apache_benchmark-P5 | Time per request:     13.479 [ms] (mean)
apache_benchmark-P5 | Time per request:     0.135 [ms] (mean, across all concurrent requests)
apache_benchmark-P5 | Transfer rate:         1963.38 [Kbytes/sec] received
apache_benchmark-P5 |
apache_benchmark-P5 | Connection Times (ms)
apache_benchmark-P5 |      min  mean[+/-sd] median  max
apache_benchmark-P5 | Connect:    0    0   0.3      0    4
apache_benchmark-P5 | Processing:  2   13   0.9     13   18
apache_benchmark-P5 | Waiting:    1   13   0.9     13   18
apache_benchmark-P5 | Total:      5   13   0.7     13   18
apache_benchmark-P5 |
apache_benchmark-P5 | Percentage of the requests served within a certain time (ms)
apache_benchmark-P5 |  50%    13
apache_benchmark-P5 |  66%    14
apache_benchmark-P5 |  75%    14
apache_benchmark-P5 |  80%    14
apache_benchmark-P5 |  90%    14
apache_benchmark-P5 |  95%    15
apache_benchmark-P5 |  98%    15
apache_benchmark-P5 |  99%    15
apache_benchmark-P5 | 100%    18 (longest request)
apache_benchmark-P5 | exited with code 0

```

```

parrot: ~/.../P5/P5-ab
- docker compose up
[+] Running 1/0
  ✓ Container apache_benchmark-P5 Created
Attaching to apache_benchmark-P5
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/1.25.4
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:      88 bytes
apache_benchmark-P5 |
apache_benchmark-P5 | Concurrency Level:    100
apache_benchmark-P5 | Time taken for tests:  9.583 seconds
apache_benchmark-P5 | Complete requests:    10000
apache_benchmark-P5 | Failed requests:      0
apache_benchmark-P5 | Total transferred:    2710000 bytes
apache_benchmark-P5 | HTML transferred:    880000 bytes
apache_benchmark-P5 | Requests per second:  1043.54 [#/sec] (mean)
apache_benchmark-P5 | Time per request:     95.827 [ms] (mean)
apache_benchmark-P5 | Time per request:     0.958 [ms] (mean, across all concurrent requests)
apache_benchmark-P5 | Transfer rate:         276.17 [Kbytes/sec] received
apache_benchmark-P5 |

```

```

apache_benchmark-P5 |
apache_benchmark-P5 | Connection Times (ms)
apache_benchmark-P5 |      min      mean[+/-sd]  median    max
apache_benchmark-P5 | Connect:    2    90    7.5    90    109
apache_benchmark-P5 | Processing:  3     6   31.6     5   3116
apache_benchmark-P5 | Waiting:    1     6   31.6     5   3116
apache_benchmark-P5 | Total:      8    95   30.8    95   3133
apache_benchmark-P5 |
apache_benchmark-P5 | Percentage of the requests served within a certain time (ms)
apache_benchmark-P5 |      50%    95
apache_benchmark-P5 |      66%    95
apache_benchmark-P5 |      75%    96
apache_benchmark-P5 |      80%    96
apache_benchmark-P5 |      90%    97
apache_benchmark-P5 |      95%    97
apache_benchmark-P5 |      98%   102
apache_benchmark-P5 |      99%   106
apache_benchmark-P5 |     100%  3133 (longest request)
apache_benchmark-P5 | exited with code 0

```

B4.2 EJECUCIÓN DE LOCUST

He tenido que hacer una pequeña modificación porque me daba el siguiente error al lanzar el docker compose de locust:

```

✓ Container p5-locust-master-mariorgzlpz-1 Created
✓ Container p5-locust-worker-mariorgzlpz-3 Recreated
✓ Container p5-locust-worker-mariorgzlpz-4 Recreated
✓ Container p5-locust-worker-mariorgzlpz-1 Recreated
✓ Container p5-locust-worker-mariorgzlpz-5 Recreated
✓ Container p5-locust-worker-mariorgzlpz-2 Recreated
Attaching to master-mariorgzlpz-1, worker-mariorgzlpz-2, worker-mariorgzlpz-3, worker-mariorgzlpz-4, worker-mariorgzlpz-5
master-mariorgzlpz-1 | [2024-05-29 21:14:35.548] c8bbd165e089/INFO/locust.main: Starting web interface at http://0.0.0.0:8089
master-mariorgzlpz-1 | [2024-05-29 21:14:35.556] c8bbd165e089/INFO/locust.main: Starting Locust 2.28.0
worker-mariorgzlpz-5 | locust: error: unrecognized arguments: -worker -master-host
worker-mariorgzlpz-5 exited with code 2
worker-mariorgzlpz-1 | locust: error: unrecognized arguments: -worker -master-host
worker-mariorgzlpz-4 | locust: error: unrecognized arguments: -worker -master-host
worker-mariorgzlpz-1 exited with code 2
worker-mariorgzlpz-3 | locust: error: unrecognized arguments: -worker -master-host
worker-mariorgzlpz-4 exited with code 2
worker-mariorgzlpz-2 | locust: error: unrecognized arguments: -worker -master-host

```

Para esto le he preguntado a ChatGPT y me ha quedado el siguiente docker compose.

```


services:
  master-mariorgzlpz:
    image: locustio/locust
    ports:
      - "8089:8089"
    volumes:
      - ./:/mnt/locust
    command: -f /mnt/locust/locustfile.py --master -H https://192.168.10.50:443/
    networks:
      p5-granjaweb_red_web:
        ipv4_address: 192.168.10.70

  worker-mariorgzlpz:
    image: locustio/locust
    volumes:
      - ./:/mnt/locust
    command: -f /mnt/locust/locustfile.py --worker --master-host=master-mariorgzlpz
    depends_on:
      - master-mariorgzlpz
    deploy:
      replicas: 5
    networks:
      p5-granjaweb_red_web:

networks:
  p5-granjaweb_red_web:
    external: true

```

Lanzando ahora el docker compose y accediendo a la direccion localhost:8089 vemos lo siguiente.


Locust

HOST
https://192.168.10.50:443/
STATUS
READY
WORKERS
5
RPS
0
FAILURES
0%

Start new load test

Numbers of users (peak concurrency)
10000

Ramp up (users started/second)
100

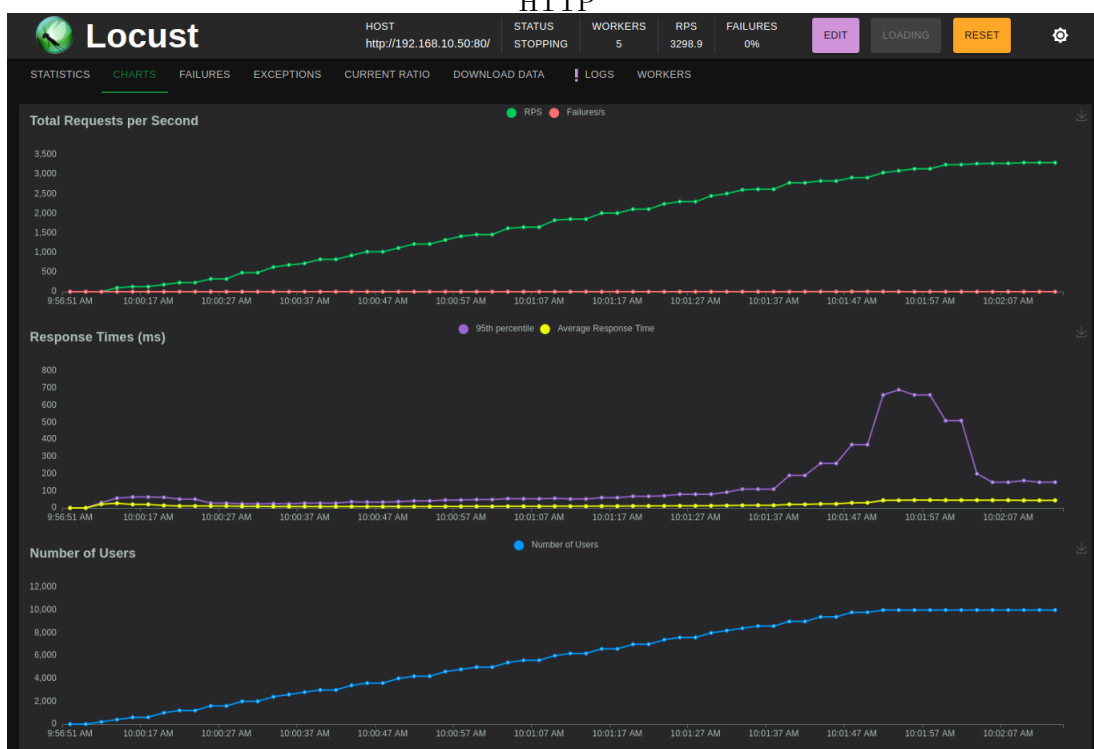
Host
https://192.168.10.50:443/


Advanced options

START

Este test de carga es para ver cuando se llega a 10000 usuarios en concurrencia. Ahora muestro los resultados obtenidos tanto en http como en https.

HTTP




Locust

HOST
http://192.168.10.50:80/
STATUS
STOPPING
WORKERS
5
RPS
3298.9
FAILURES
0%

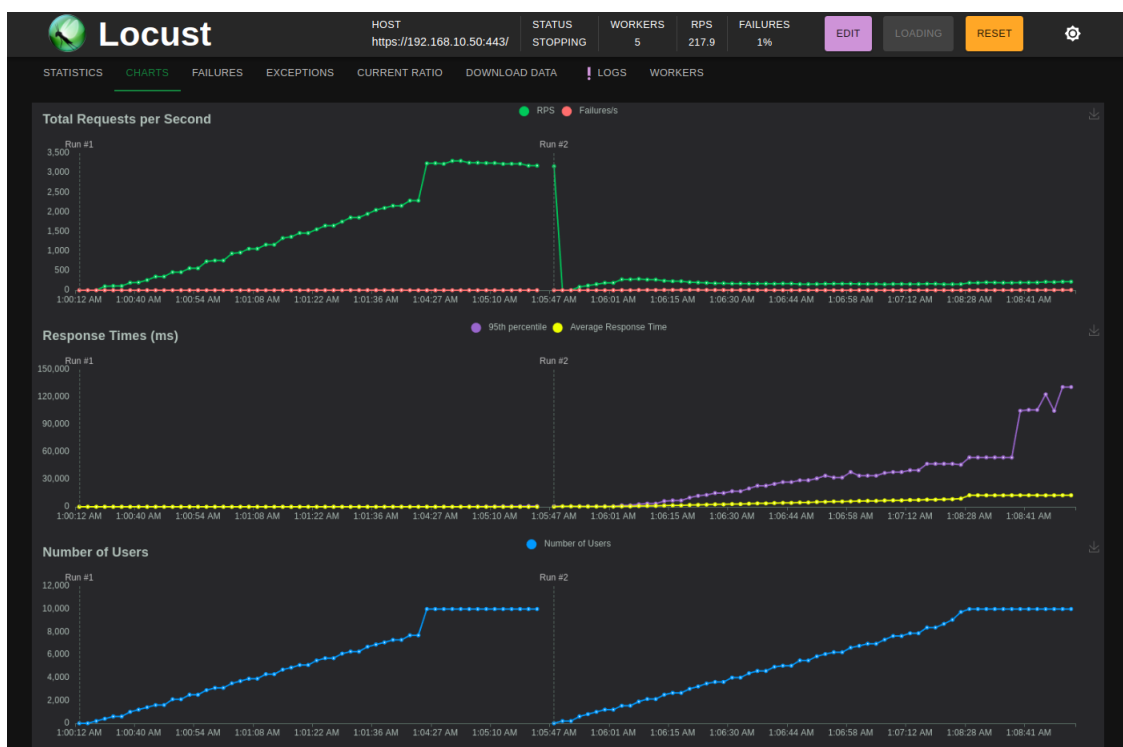
STATISTICS
CHARTS
FAILURES
EXCEPTIONS
CURRENT RATIO
DOWNLOAD DATA
LOGS
WORKERS

Type	Name	# Requests	# Fails	Median (ms)	95%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
GET	/index.php	234840	52	9	170	630	44.06	1	1662	87.98	3298.9	0
	Aggregated	234840	52	9	170	630	44.06	1	1662	87.98	3298.9	0


HTTPS



Además para comparar los dos ejecuto uno despues de otro(primero http y luego https) con lo que me queda el siguiente resultado:



Debajo muestro los logs y failures de locust donde podemos ver que los workers llegan a la capacidad máxima de la cpu provocando algunos de los saltos y los fallos que vemos en las imágenes.

**Locust**

HOST
http://192.168.10.50:80/

STATUS
STOPPING

WORKERS
5


RPS
3298.9

FAILURES
0%

EDIT

LOADING


RESET



STATISTICS CHARTS FAILURES EXCEPTIONS CURRENT RATIO DOWNLOAD DATA LOGS WORKERS

Logs

- [2024-06-01 07:56:41.787] c8bbd165e089/INFO/locust.main: Starting web interface at http://0.0.0.0:8089
- [2024-06-01 07:56:41.802] c8bbd165e089/INFO/locust.main: Starting Locust 2.28.0
- [2024-06-01 07:56:42.015] c8bbd165e089/INFO/locust.runners: Worker 95a62627d6bc_a20a655137ff4aa09cd64fe88ebf4263 (index 0) reported as ready. 1 workers connected.
- [2024-06-01 07:56:42.371] c8bbd165e089/INFO/locust.runners: Worker 369d66fe92a5_0680d4a928aa46a99c1e2ffa4b450a3d (index 1) reported as ready. 2 workers connected.
- [2024-06-01 07:56:42.646] c8bbd165e089/INFO/locust.runners: Worker b4243fc7735a_8b02c44d880e403481d4dc9a625ce39c (index 2) reported as ready. 3 workers connected.
- [2024-06-01 07:56:42.847] c8bbd165e089/INFO/locust.runners: Worker 618f1fb19de7_39b1531d458d4f7ab98388443b496355 (index 3) reported as ready. 4 workers connected.
- [2024-06-01 07:56:42.993] c8bbd165e089/INFO/locust.runners: Worker d6ce53157661_a35800a8ed4a4bafbec497be8145dd7e (index 4) reported as ready. 5 workers connected.
- [2024-06-01 08:00:07.316] c8bbd165e089/INFO/locust.runners: Sending spawn jobs of 10000 users at 100.00 spawn rate to 5 ready workers
- [2024-06-01 08:01:37.503] c8bbd165e089/WARNING/locust.runners: Worker 95a62627d6bc_a20a655137ff4aa09cd64fe88ebf4263 (index 0) exceeded cpu threshold (will only log this once per worker)
- [2024-06-01 08:01:38.105] c8bbd165e089/WARNING/locust.runners: Worker b4243fc7735a_8b02c44d880e403481d4dc9a625ce39c (index 2) exceeded cpu threshold (will only log this once per worker)
- [2024-06-01 08:01:38.359] c8bbd165e089/WARNING/locust.runners: Worker d6ce53157661_a35800a8ed4a4bafbec497be8145dd7e (index 4) exceeded cpu threshold (will only log this once per worker)
- [2024-06-01 08:01:38.486] c8bbd165e089/WARNING/locust.runners: Worker 618f1fb19de7_39b1531d458d4f7ab98388443b496355 (index 3) exceeded cpu threshold (will only log this once per worker)
- [2024-06-01 08:01:42.848] c8bbd165e089/WARNING/locust.runners: Worker 369d66fe92a5_0680d4a928aa46a99c1e2ffa4b450a3d (index 1) exceeded cpu threshold (will only log this once per worker)
- [2024-06-01 08:01:46.637] c8bbd165e089/INFO/locust.runners: All users spawned: ("P5_usuarios": 10000) (10000 total users)

**Locust**

HOST
http://192.168.10.50:80/

STATUS
STOPPING

WORKERS
5


RPS
3298.9

FAILURES
0%

EDIT


LOADING

RESET



STATISTICS CHARTS FAILURES EXCEPTIONS CURRENT RATIO DOWNLOAD DATA LOGS WORKERS

# Failures	Method	Name	Message
1	GET	/index.php	ConnectionResetError(104, 'Connection reset by peer')
51	GET	/index.php	RemoteDisconnected('Remote end closed connection without response')

**Locust**

HOST
https://192.168.10.50:443/

STATUS
STOPPING

WORKERS
5


RPS
171

FAILURES
1%

EDIT

LOADING


RESET



STATISTICS CHARTS FAILURES EXCEPTIONS CURRENT RATIO DOWNLOAD DATA LOGS WORKERS

Logs

- [2024-06-01 08:14:11.099] c8bbd165e089/INFO/locust.main: Starting web interface at http://0.0.0.0:8089
- [2024-06-01 08:14:11.106] c8bbd165e089/INFO/locust.main: Starting Locust 2.28.0
- [2024-06-01 08:14:11.425] c8bbd165e089/INFO/locust.runners: Worker 95a62627d6bc_55f484c0b2524b1a9bef42de15abce7d (index 0) reported as ready. 1 workers connected.
- [2024-06-01 08:14:11.733] c8bbd165e089/INFO/locust.runners: Worker 369d66fe92a5_de24122a6969474ab8b5e2e815c1379b (index 1) reported as ready. 2 workers connected.
- [2024-06-01 08:14:12.054] c8bbd165e089/INFO/locust.runners: Worker b4243fc7735a_1697115fabfd4b4f869f89b6ce57d04b (index 2) reported as ready. 3 workers connected.
- [2024-06-01 08:14:12.351] c8bbd165e089/INFO/locust.runners: Worker 618f1fb19de7_8c62ada38aa547d5b005857ae2545cee (index 3) reported as ready. 4 workers connected.
- [2024-06-01 08:14:12.582] c8bbd165e089/INFO/locust.runners: Worker d6ce53157661_ec277a6c0ad14c638ee52d67643045c0 (index 4) reported as ready. 5 workers connected.
- [2024-06-01 08:14:20.880] c8bbd165e089/INFO/locust.runners: Sending spawn jobs of 10000 users at 100.00 spawn rate to 5 ready workers
- [2024-06-01 08:14:37.103] c8bbd165e089/WARNING/locust.runners: Worker 95a62627d6bc_55f484c0b2524b1a9bef42de15abce7d (index 0) exceeded cpu threshold (will only log this once per worker)
- [2024-06-01 08:14:37.610] c8bbd165e089/WARNING/locust.runners: Worker 369d66fe92a5_de24122a6969474ab8b5e2e815c1379b (index 1) exceeded cpu threshold (will only log this once per worker)
- [2024-06-01 08:14:38.050] c8bbd165e089/WARNING/locust.runners: Worker b4243fc7735a_1697115fabfd4b4f869f89b6ce57d04b (index 2) exceeded cpu threshold (will only log this once per worker)
- [2024-06-01 08:14:38.095] c8bbd165e089/WARNING/locust.runners: Worker 618f1fb19de7_8c62ada38aa547d5b005857ae2545cee (index 3) exceeded cpu threshold (will only log this once per worker)
- [2024-06-01 08:14:38.596] c8bbd165e089/WARNING/locust.runners: Worker d6ce53157661_ec277a6c0ad14c638ee52d67643045c0 (index 4) exceeded cpu threshold (will only log this once per worker)
- [2024-06-01 08:16:00.952] c8bbd165e089/INFO/locust.runners: Spawning is complete and report waittime is expired, but not all reports received from workers: ("P5_usuarios": 9540) (9540 total users)

**Locust**

HOST
https://192.168.10.50:443/

STATUS
STOPPING

WORKERS
5


RPS
171

FAILURES
1%

EDIT

LOADING

RESET



STATISTICS CHARTS FAILURES EXCEPTIONS CURRENT RATIO DOWNLOAD DATA LOGS WORKERS

# Failures	Method	Name	Message
199	GET	/index.php	RemoteDisconnected('Remote end closed connection without response')

B4.3 Analisis Propuesta IA

Preguntas sobre Apache Benchmark Le pregunto a ChatGPT por el error:


```
docker compose up
[+] Running 1/0
✓ Container apache_benchmark-P5 Created 0.1s
Attaching to apache_benchmark-P5
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 | SSL read failed (5) - closing connection
apache_benchmark-P5 | SSL read failed (5) - closing connection
```



The error message **SSL read failed (5) - closing connection** in ApacheBench (ab) indicates that there is an issue with the SSL/TLS handshake or the SSL connection itself. This could be due to several reasons, such as misconfigured SSL settings, issues with the server's SSL certificate, or network problems.

Here are some steps to troubleshoot and resolve this issue:

1. Check SSL/TLS Configuration

Ensure that the server you are benchmarking has a valid SSL/TLS certificate and that it is properly configured. You can use tools like **openssl** to test the connection:

```
sh Copy code
openssl s_client -connect 192.168.10.50:443
```

This command will try to establish an SSL connection to the specified server and port. Look for any errors or warnings in the output.

2. Verify Certificate Validity

Make sure the SSL certificate is not expired and is correctly installed. You can use online tools like SSL Labs' SSL Test or **openssl** for local verification:

```
sh Copy code
openssl x509 -in /path/to/certificate.crt -noout -dates
```

This will display the start and end dates of the certificate validity.

3. Update ApacheBench

Ensure you are using the latest version of ApacheBench, as older versions might have issues with newer SSL/TLS protocols. If you installed it via a package manager, you might want to update it:

```
sh Copy code
sudo apt-get update
sudo apt-get install apache2-utils
```

Or, if you are using another OS, update the package accordingly.

4. Disable SSL Verification (For Testing Purposes Only)

If you suspect that the issue might be related to strict SSL verification, you can temporarily disable SSL verification in ApacheBench to see if the issue persists. Note that this is only for testing purposes and should not be used in production:

```
sh Copy code  
ab -k -n 100 -c 10 -Z /path/to/ca-bundle.crt https://192.168.10.50/
```

5. Check Server Logs

Look at the server logs for any SSL-related errors or warnings. This might provide additional insights into what is going wrong.

6. Test with a Different Tool

To confirm that the issue is with ApacheBench, you can try benchmarking with another tool like `curl` or `wrk`:

```
sh Copy code  
curl -v https://192.168.10.50
```

If `curl` also fails with SSL errors, it indicates a server-side issue.

7. Network Issues

Ensure there are no network issues between your client and the server. Sometimes intermediate proxies or firewalls might cause SSL/TLS handshake failures.

Summary

By following these steps, you can systematically identify and resolve the SSL read failure issues in ApacheBench. The most common problems are typically related to SSL configuration or certificate validity, so start by verifying those aspects. ↓

mi certificado es autofirmado



Si estás utilizando un certificado autofirmado, es posible que ApacheBench esté fallando debido a la falta de confianza en el certificado. Aquí hay algunas formas de solucionar este problema:

1. Añadir el Certificado Autofirmado al Cliente ApacheBench

Puedes agregar el certificado autofirmado a la lista de certificados de confianza en tu sistema. Para hacer esto:

1. **Exporta el certificado autofirmado** si no lo tienes ya en un archivo `.crt`.
2. **Añade el certificado a los certificados de confianza de tu sistema.**

En Linux, puedes hacer esto copiando el archivo `.crt` a `/usr/local/share/ca-certificates/` y luego actualizando los certificados:

```
sh Copy code  
sudo cp /ruta/a/tu/certificado.crt /usr/local/share/ca-certificates/  
sudo update-ca-certificates
```

Preguntas sobre locust Le he preguntado a ChatGPT por el error lo que me ha dado la siguiente respuesta:

The error message ``Unknown User(s): -worker, -master-host, master-mariorgzlpz`` suggests that there is an issue with how the command-line arguments are being passed to the Locust workers. It seems like the arguments for the worker nodes might be incorrectly formatted or there might be an issue with how the dashes (```) are encoded.

Here are some steps to troubleshoot and fix the issue:

1. **Check for Encoding Issues:**

Ensure that the dashes are standard hyphens (`-`) and not en-dashes (`-`). This is a common issue when copying and pasting commands from certain text editors or documents.

2. **Correct Command-Line Arguments:**

Verify the command-line arguments being passed to the workers. They should look something like this:

```
sh Copy code  
locust -worker --master-host=master-mariorgzlpz
```

Las respuestas que me ha dado ChatGPT han sido validas y han hecho que me funcionen las dos herramientas de testing.

B5.ANALISIS DE RESULTADOS

B5.1 ANÁLISIS APACHE BENCHMARK

Lo primero será explicar las diferentes métricas y las más relevantes que podemos extraer de apache benchmark. Toda la información ha sido extraída de la [documentación](#) oficial de apache benchmark

Métricas importantes

- **Time taken for test** Este es el tiempo tomado desde el momento en que se crea la primera conexión de socket hasta el momento en que se recibe la última respuesta.
- **Completed/Failed request** El número de respuestas exitosas/fallidas.
- **Total/HTML transferred** Total/HTML transferido en bytes.
- **Transfer Rate** La tasa de transferencia calculada con la fórmula $\text{totalread} / 1024 / \text{timetaken}$.
- **Connection times** Indica el tiempo que tarda en establecerse una conexión entre un cliente y un servidor.
- Hay muchas metricas (request per second , time per request) que se calculan como operaciones de otras metricas por tanto esas no las tendremos en cuenta (Ya que las comparativas serían iguales)

-Las claras diferencias que podemos ver son por ejemplo el tiempo tomado para la prueba que en http es de 1.348 segundos mientras que en https es de 9.583 lo que supone un aumento de unas 7 veces en el tiempo. Esto es debido al uso del certificado y al tener que realizar las comprobaciones de seguridad. En general http no tiene por que ser más rapido que https pero en este caso por la configuración tan simple tenemos que si ocurre esto.

-También podemos ver que en la imagen https nos aparece los cifrados que usamos y el TLS empleado algo a tener en cuenta también a la hora de comparar y extraer métricas puesto que es mejor tener mayor nivel de seguridad aunque pueda ir más lento.

-Vemos que tanto en http como en https el numero de respuestas exitosas es de 10000 lo que indica que todas las request hechas han sido recibidas. Además podemos ver que el numero de bytes totales transferidos es el mismo debido a que ambos realizan la misma operación get sobre el mismo index.php. -Claramente la tasa de transferencia(kbytes enviados/ tiempo) será superior en el caso de http puesto que para calcularla se emplea el tiempo total y puesto que http es más rapido también será mayor su tasa de transferencia.

-Comparando ahora connection times podemos ver que http vuelve a ser más rápido que https en el tiempo mínimo que en el caso de http es de 13ms y en el caso de https es de 95ms.

B5.2 ANÁLISIS LOCUST

Igual que para apache comenzaremos explicando las diferentes métricas de locust. Las explicaciones del análisis han sido extraídas de la siguiente [página](#)

Métricas importantes

- **Total Requests per Second (RPS)** El número de solicitudes por segundo que la aplicación puede manejar.
- **Average Response Times (ms)** El tiempo de respuesta promedio de todas las solicitudes.
- **Number of Users** El número de usuarios concurrentes que la aplicación puede manejar.
- **Percentil 95** Representa el tiempo de respuesta que el 95 % de las solicitudes completaron en menos tiempo que ese valor.
- **Failures** Representan el número de solicitudes que no se completaron correctamente.

Viendo la imagen de la comparativa (el primer gráfico es http y el segundo https) nos damos cuenta de que http es más rápido que https, simplemente comparando las métricas extraídas en cada caso. Veamoslas una por una (Para esto es mejor ver las imágenes separadas de cada una):

-Tiempo que tardas en llegar a 10000 usuarios en cada caso: Ambos casos tarda practicamente lo mismo unos 2 minutos. Esto tiene sentido puesto que le hemos puesto que aumente a un ritmo de 100/s.

-Como vemos el numero de request por segundo es muy superior en http viendo como alcanza un pico de unas 3000 request mientras que https apenas alcanza los 300. Esto también se ve reflejado en el número de peticiones recibidas siendo el de http un total de 234840 mientras que en https se alcanzan 23156 unas 10 veces menos peticiones recibidas en el mismo tiempo.

-El tiempo de respuesta promedio (gráfica amarilla) de http es mucho más bajo (un average de 44 ms) lo que nos da una idea de porque el numero request por segundo es mayor. En https esta misma métrica nos arroja un promedio de 10000 ms siendo bastante superior.

-El percentil 95 nos puede ayudar a identificar problemas de rendimiento que afectan a una pequeña porción de usuarios. Sabiendo esto, podemos ver que en http es practicamente igual que el tiempo promedio de respuesta casi todo el tiempo, lo que nos indica que casi todo el mundo tiene el mismo tiempo de respuesta que es un buen indicativo. Por otra parte, en https vemos como si se diferencia bastante el tiempo de respuesta del percentil 95 lo que nos indica que el servidor no gestiona tan bien un numero alto de peticiones.

-El numero de usuarios es el mismo al final puesto que lo habiamos decidido asi al principio.

-Podemos ver que el numero de fallos en http es de 52 un numero insignificante para el total de peticiones(230000) que se habían mandado. Además los fallos son debidos a que los workers llegan al máximo de CPU y arroja el fallo RemoteDisconnected('Remote end closed connection without response') indicando que no se ha obtenido respuesta. También nos sale una vez el fallo ConnectionResetError(104, 'connection reset by peer') que es debido a que el servidor ha llegado al limite de peticiones por segundo [explicación aquí](#). En https nos salen un total de 199 fallos, todas el mismo error de RemoteDisconnected y provocado por que los workers alcanzan el limite de CPU, que para el total de peticiones(23000) supone aproximadamente un 1 % algo que podría suponer un problema si aumenta un poco más.

Conclusión

Viendo ambos test de carga es claro que http ofrece mejores tiempos de respuesta y tiene mayor tolerancia a fallos aunque debemos tener en cuenta la seguridad que nos ofrece https.

A1.DESARROLLAR TAREAS AVANZADAS EN LOCUSTFILE.PY

A1.1 Propuesta final

Me queda el siguiente locustfile.py:

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

class P5_tuusuariougr(TaskSet):

    def on_start(self):
        """ Autenticacion de usuarios al inicio de la prueba """
```

```

        self.login()

def login(self):
    """Simula el inicio de sesion de un usuario"""
    self.client.post("/wp-login.php", {"log": "testuser", "pwd": "password"}, verify=False)

@task(1)
def load_index(self):
    """Carga la pagina principal"""
    self.client.get("/index.php", verify=False)

@task(2)
def load_page(self):
    """Carga una pagina de contenido"""
    self.client.get("/wp-load.php", verify=False)

@task(3)
def create_post(self):
    """Simula la creacion de una nueva publicacin"""
    self.client.post("/create-post.php", {"title": "Nuevo post", "content": "
Contenido del post"}, verify=False)

@task(4)
def insert_comment(self):
    """Simula la insercion de un comentario en una publicacion"""
    self.client.post("/wp-comment-post.php", {"post_id": 1, "comment": "Este es un
comentario"}, verify=False)

@task(5)
def search(self):
    """Simula una consulta de busqueda"""
    self.client.get("/search.php?q=locust", verify=False)

@task(6)
def logout(self):
    """Simula el cierre de sesion de un usuario"""
    self.client.get("/logout.php", verify=False)

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

```

Con ese fichero realizo demasiadas tareas por lo que decido comentar algunas de ellas en la prueba para poder apreciar mejor en el test de carga. Este es el fichero que uso finalmente en el test de carga:

```

from locust import HttpUser, TaskSet, task, between

class P5_tuusuariougr(TaskSet):

    # def on_start(self):
    #     """Autenticacion de usuarios al inicio de la prueba"""
    #     self.login()

    @task(1)
    def load_index(self):
        """Carga la pagina principal"""
        self.client.get("/index.php", verify=False)

    @task(2)
    def load_page(self):
        """Carga una pagina de contenido"""
        self.client.get("/wp-load.php", verify=False)

    @task(3)
    def login(self):
        """Simula el inicio de sesion de un usuario"""
        self.client.post("/wp-login.php", {"log": "testuser", "pwd": "password"}, verify=False)

    # def create_post(self):
    #     """Simula la creacion de una nueva publicacion"""
    #     self.client.post("/create-post.php", {"title": "Nuevo post", "content": "

```

```

        Contenido del post"}, verify=False)

@task(4)
def insert_comment(self):
    """Simula la insercion de un comentario en una publicacion"""
    self.client.post("/wp-comment-post.php", {"post_id": 1, "comment": "Este es un
comentario"}, verify=False)

# @task(5)
# def search(self):
#     """Simula una consulta de busqueda"""
#     self.client.get("/search.php?q=locust", verify=False)

# @task(6)
# def logout(self):
#     """Simula el cierre de sesion de un usuario"""
#     self.client.get("/logout.php", verify=False)

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

```

A1.2 Analisis Propuesta IA

Pregunta y respuesta ChatGPT

Tengo el siguiente 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)

```

Quiero hacer lo siguiente:

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.



Para desarrollar tareas más avanzadas en tu `locustfile.py` y reflejar operaciones típicas de un CMS, puedes añadir las siguientes funcionalidades:

1. **Navegación por páginas:** Puedes simular la navegación a diferentes páginas de tu sitio web.
2. **Creación o interacción con contenido:** Simula la creación de contenido, como publicaciones o comentarios.
3. **Interacciones con la base de datos:** Inserta comentarios o publicaciones.
4. **Autenticación de usuarios:** Simula el inicio de sesión de los usuarios.
5. **Carga de múltiples tipos de contenido:** Simula la carga de diferentes tipos de contenido, como artículos, imágenes, etc.
6. **Ejecución de consultas de búsqueda:** Simula la ejecución de consultas de búsqueda.

A continuación, se muestra un ejemplo de cómo se podría estructurar un archivo `locustfile.py` para cumplir con estos requisitos:

Analisis Lo que me proporciona ChatGPT me parece una posible implementación de lo que yo quiero hacer por lo que tomando esto como base realizo las modificaciones para realizar algunas tareas avanzadas con locust y probar el test. Simplemente modifico las URLs y uso las tareas que me parezcan convenientes sobre mi web de wordpress.

A2.CREAR ESCENARIO MULTICONTENEDOR CON ALGÚN CMS

A2.1 Propuesta final

Finalmente me queda el siguiente docker compose, al cual he llamado docker-compose-wordpress.yml :

```
version: '4.0'

services:
  web1:s
    image: wordpress:latest
    restart: always
    volumes:
      - ./web_wordpress_MarioRgzLpz:/var/www/html
      - ./P4-MarioRgzLpz-apache/MarioRgzLpz-apache-ssl.conf:/etc/apache2/conf.d/MarioRgzLpz-apache-ssl.conf
      - ./P4-MarioRgzLpz-certificados:/etc/apache2/ssl
    environment:
      WORDPRESS_DB_HOST: mysql
      WORDPRESS_DB_USER: miusuario
      WORDPRESS_DB_PASSWORD: mipassword
      WORDPRESS_DB_NAME: wordpress
    container_name: web1
    networks:
      red_web:
        ipv4_address: 192.168.10.2
      red_servicios:
        ipv4_address: 192.168.20.2
    cap_add:
      - NET_ADMIN
    depends_on:
      - mysql

  web2:
    image: wordpress:latest
    restart: always
    volumes:
      - ./web_wordpress_MarioRgzLpz:/var/www/html
      - ./P4-MarioRgzLpz-apache/MarioRgzLpz-apache-ssl.conf:/etc/apache2/conf.d/MarioRgzLpz-apache-ssl.conf
      - ./P4-MarioRgzLpz-certificados:/etc/apache2/ssl
    environment:
      WORDPRESS_DB_HOST: mysql
      WORDPRESS_DB_USER: miusuario
      WORDPRESS_DB_PASSWORD: mipassword
      WORDPRESS_DB_NAME: wordpress
    container_name: web2
    networks:
      red_web:
        ipv4_address: 192.168.10.3
      red_servicios:
        ipv4_address: 192.168.20.3
    cap_add:
      - NET_ADMIN
    depends_on:
      - mysql
    #Resto de contenedores web3,web4 ...

  balanceador-nginx:
    image: mariorgzlpz-nginx-image:p4
    command: ['nginx', '-g', 'daemon off;']
    restart: always
    volumes:
      - ./P4-MarioRgzLpz-nginx/MarioRgzLpz-nginx-ssl.conf:/etc/nginx/nginx.conf
      - ./P4-MarioRgzLpz-certificados/certificado_MarioRgzLpz.crt:/etc/nginx/ssl/certificado_MarioRgzLpz.crt
      - ./P4-MarioRgzLpz-certificados/certificado_MarioRgzLpz.key:/etc/nginx/ssl/certificado_MarioRgzLpz.key
      - ./ticket_sesion_MarioRgzLpz.key:/etc/nginx/ssl/session_ticket.key
    container_name: balanceador-nginx
    ports:
```

```

    - 80:80
    - 443:443
networks:
  red_web:
    ipv4_address: 192.168.10.50
cap_add:
  - NET_ADMIN
depends_on:
  - web1
  - web2
  - web3
  - web4
  - web5
  - web6
  - web7
  - web8
  - mysql

mysql:
  image: mysql:5.7
  volumes:
    - ./mysql_data_MarioRgzLpz:/var/lib/mysql
  container_name: base_datos
  tty: true
  restart: always
  environment:
    MYSQL_ROOT_PASSWORD: 1234
    MYSQL_DATABASE: wordpress
    MYSQL_USER: miusuario
    MYSQL_PASSWORD: mipassword
  networks:
    red_servicios:
      ipv4_address: 192.168.20.11

networks:
  red_web:
    driver: bridge
    ipam:
      config:
        - subnet: 192.168.10.0/24
          gateway: 192.168.10.1
  red_servicios:
    driver: bridge
    ipam:
      config:
        - subnet: 192.168.20.0/24
          gateway: 192.168.20.1

```

Para el cual he añadido el servicio de mysql además de cambiar la imagen de mis servicios web por la de wordpress. También he añadido un volumen a la base de datos, y configurado las variables de entorno de los servicios de wordpress y de la base de datos. Todo esto realizado siguiendo el siguiente [blog](#). La base de datos estará conectada a la red servicios. Además he tenido que modificar la configuración de nginx siguiendo lo que me dice ChatGPT en la respuesta porque si no me daba error al acceder a ciertas páginas. Esto me queda como sigue:

```

events {
    worker_connections 1024;
}

http {
    # Configuración de la cache de sesiones SSL y tickets de sesión
    ssl_session_cache shared:SSL:10m;
    ssl_session_timeout 10m;
    # Limitamos el número de conexiones por IP
    # limit_conn_zone $binary_remote_addr zone=prueba:10m;
    # limit_req_zone $binary_remote_addr zone=one:10m rate=1r/s;

    # add_header X-Frame-Options SAMEORIGIN;
    # add_header X-XSS-Protection "1; mode=block";
    # add_header Content-Security-Policy "default-src 'self' http: https: data: blob: '
    unsafe-inline'" always;
    # add_header X-Content-Type-Options nosniff;

```



```

upstream localhost { #backend_MarioRgzLpz
    server 192.168.10.2 weight=1;
    server 192.168.10.3 weight=1;
    server 192.168.10.4 weight=1;
    server 192.168.10.5 weight=1;
    server 192.168.10.6 weight=1;
    server 192.168.10.7 weight=1;
    server 192.168.10.8 weight=1;
    server 192.168.10.9 weight=1;
}

server {
    listen 80;
    server_name nginx-mariorgzlpz;

    http2 on;

    access_log /var/log/nginx/nginx-mariorgzlpz.access_log;
    error_log /var/log/nginx/nginx-mariorgzlpz.error_log;

    location / {
        proxy_pass http://localhost; #http://backend_MarioRgzLpz
        proxy_set_header Cookie $http_cookie;
        proxy_hide_header Set-Cookie;
        # deny all;
    }

    location /estadisticas-mariorgzlpz {
        stub_status on;
    }
}

server {
    listen 443 ssl;
    server_name nginx-mariorgzlpz;

    http2 on;

    # Activamos los tickets de sesion y cargamos nuestro ticket
    ssl_session_tickets on;
    ssl_session_ticket_key /etc/nginx/ssl/session-ticket.key;

    # Configuracion de parametros SSL adicionales
    ssl_protocols TLSv1.2 TLSv1.3; #Eleccion de protocolos SSL
    ssl_prefer_server_ciphers on; # Preferencia de la lista de cifrados del servidor
    sobre la del cliente
    ssl_ciphers 'EECDH+AESGCM:EDH+AESGCM:AES256+EECDH:AES256+EDH'; # Lista de suites
    de cifrado

    ssl_certificate /etc/nginx/ssl/certificado_MarioRgzLpz.crt;
    ssl_certificate_key /etc/nginx/ssl/certificado_MarioRgzLpz.key;

    access_log /var/log/nginx/nginx-mariorgzlpz.access_log;
    error_log /var/log/nginx/nginx-mariorgzlpz.error_log;

    location / {
        # limit_conn prueba 4;
        # limit_req zone=one burst=2;
        proxy_pass https://localhost; #https://backend_MarioRgzLpz
        proxy_set_header Cookie $http_cookie;
        proxy_hide_header Set-Cookie;
    }

    location /estadisticas-mariorgzlpz {
        stub_status on;
    }

    # if ($http_user_agent ~* LWP::Simple|BBBike|wget) {
    #     return 403;
    # }
}
}

```

Ahora voy a mostrar lo que me va apareciendo para realizar la instalación de wordpress. Muestro que me

aparece de manera correcta tanto en http como en https :

The image displays two screenshots of the WordPress installation process.

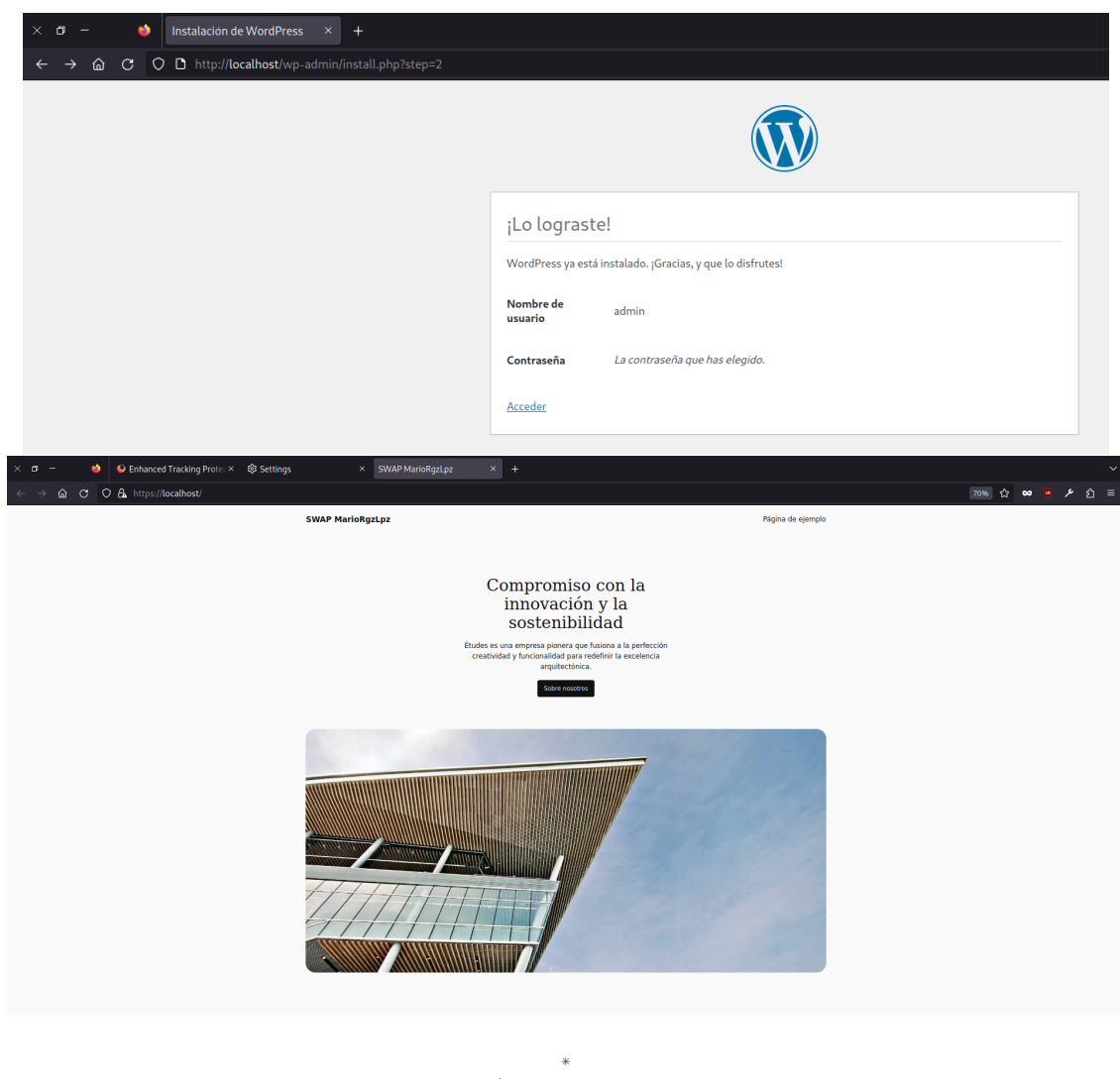
The top screenshot shows the language selection screen. The URL in the browser is `http://localhost/wp-admin/install.php`. The WordPress logo is at the top. A dropdown menu is open, showing a list of languages. "English (United States)" is selected. Other visible languages include Afrikaans, Azərbaycan dili, Беларуская мова, Български, Bosanski, Català, and Cebuano. A "Continue" button is at the bottom right of the language list.

The bottom screenshot shows the "Información necesaria" (Required Information) screen. The URL in the browser is `http://localhost/wp-admin/install.php?step=1`. The WordPress logo is at the top. The page has a heading "Hola" and a subheading "¡Este es el famoso proceso de instalación de WordPress en cinco minutos! Simplemente completa la información siguiente y estarás a punto de usar la más enriquecedora y potente plataforma de publicación personal del mundo."

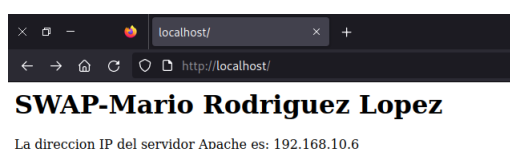
The "Información necesaria" section contains the following fields and options:

- Título del sitio:** A text input field containing "SWAP MarioRgzLpz".
- Nombre de usuario:** A text input field containing "admin". Below the field, it says: "Los nombres de usuario pueden tener únicamente caracteres alfanuméricos, espacios, guiones bajos, guiones medios, puntos y el símbolo @."
- Contraseña:** A text input field containing "admin". To the right is an "Ocultar" button. Below the field, it says "Muy débil". Below that, it says: "Importante: Necesitas esta contraseña para acceder. Por favor, guárdala en un lugar seguro."
- Confirma la contraseña:** A checkbox labeled "Confirma el uso de una contraseña débil." which is checked.
- Tu correo electrónico:** A text input field containing "MarioRgzLpz@correo.ugr.es". Below the field, it says: "Comprueba bien tu dirección de correo electrónico antes de continuar."
- Visibilidad en los motores de búsqueda:** A checkbox labeled "Pedir a los motores de búsqueda que no indexen este sitio" which is unchecked. Below it, it says: "Depende de los motores de búsqueda atender esta petición o no."

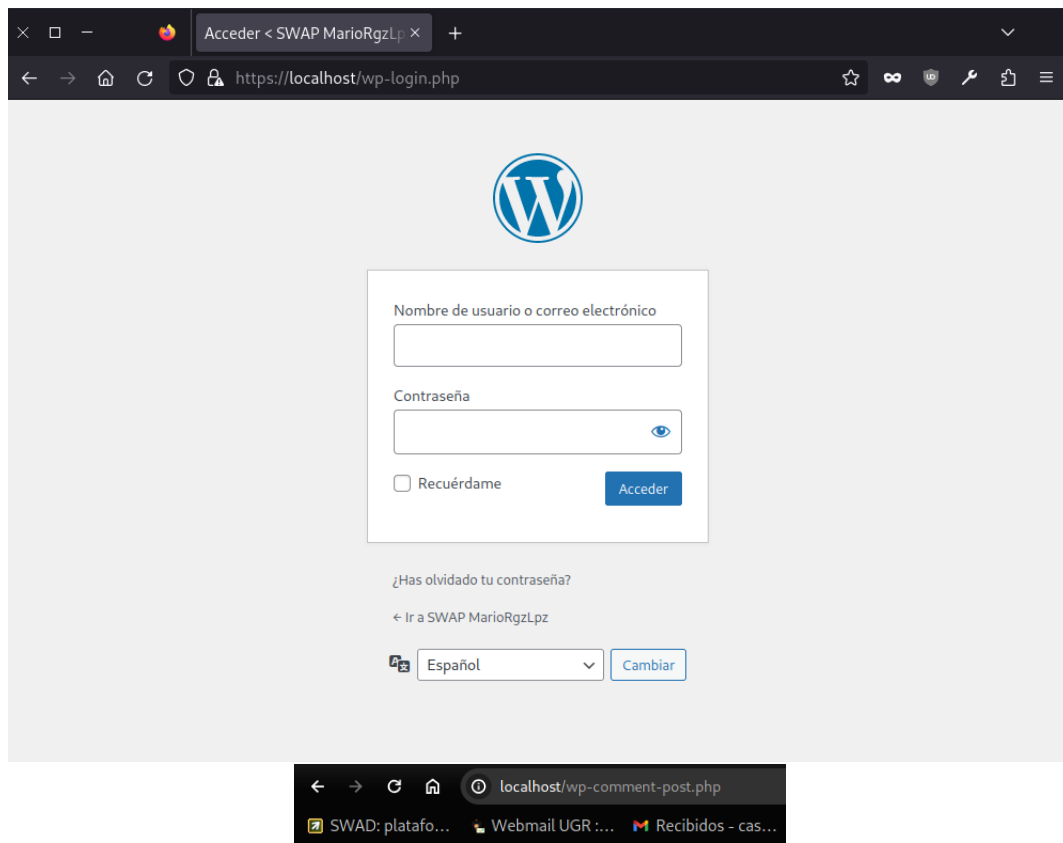
At the bottom of the form is a button labeled "Instalar WordPress".



Tras esto modifico el index.php para que en lugar de la pagina de wordpress me aparezca el index creado por nosotros. Así me queda finalmente configurado y funcionando tanto en http como en https.



Aquí muestro las páginas de login donde ejecutaremos el test de locust para probar inicio de sesión. También la dirección donde se pueden hacer comentarios.



SWAP-Mario Rodriguez Lopez

La direccion IP del servidor Apache es: 192.168.10.5

El login me daba problemas de cookies y siguiendo los pasos que me dice la propia pagina de wordpress me sigue sin funcionar aunque me permite hacer peticiones post sin dar fallo en la mayoría de los casos.

A2.2 Analisis Propuesta IA

Pregunta y respuesta ChatGPT Quiero hacer ahora lo siguiente

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

Yo tengo este docker compose para mi granja (Le paso el docker compose de la P4)

Para crear un escenario multicontenedor con un CMS (como WordPress) e integrarlo con la granja web existente, debes:

1. **Añadir contenedores para el CMS:** Crear contenedores adicionales para el CMS seleccionado.
2. **Configurar la base de datos:** Añadir un contenedor para la base de datos.
3. **Ajustar el balanceador de carga:** Configurar el balanceador de carga para redirigir el tráfico a la nuevas instancias del CMS.
4. **Conectar a las redes existentes:** Asegurarse de que todos los contenedores estén conectados a las redes adecuadas (`red_web` y `red_servicios`).

A continuación, se muestra cómo ajustar tu archivo `docker-compose.yml` para incluir WordPress como CMS:

docker-compose.yml

```
version: '4.0'

services:
  web1:
    image: mariorgzlpz-apache-image:p4
    volumes:
      - ./web_MarioRgzLpz:/var/www/localhost/htdocs
      - ./P4-MarioRgzLpz-apache/MarioRgzLpz-apache-ssl.conf:/etc/apache2/conf.d/MarioRgzLpz-apache-ssl.conf
      - ./P4-MarioRgzLpz-certificados:/etc/apache2/ssl
    container_name: web1
    networks:
      red_web:
        ipv4_address: 192.168.10.2
      red_servicios:
        ipv4_address: 192.168.20.2
    cap_add:
      - NET_ADMIN

# ... repetir para web2 a web8 ...

balanceador-nginx:
  image: mariorgzlpz-nginx-image:p4
  command: ['nginx', '-g', 'daemon off;']
  volumes:
    - ./P4-MarioRgzLpz-nginx/MarioRgzLpz-nginx-ssl.conf:/etc/nginx/nginx.conf
    - ./P4-MarioRgzLpz-certificados/certificado_MarioRgzLpz.crt:/etc/nginx/ssl/certificado_MarioRgzLpz.crt
    - ./P4-MarioRgzLpz-certificados/certificado_MarioRgzLpz.key:/etc/nginx/ssl/certificado_MarioRgzLpz.key
    - ./ticket_sesion_MarioRgzLpz.key:/etc/nginx/ssl/session_ticket.key
  container_name: balanceador-nginx
  ports:
    - 80:80
    - 443:443
  networks:
    red_web:
      ipv4_address: 192.168.10.50
  cap_add:
    - NET_ADMIN

wordpress:
  image: wordpress:latest
  volumes:
    - ./wp-content:/var/www/html/wp-content
  environment:
    WORDPRESS_DB_HOST: db
    WORDPRESS_DB_USER: wordpress
    WORDPRESS_DB_PASSWORD: wordpress
    WORDPRESS_DB_NAME: wordpress
  networks:
    red_web:
      ipv4_address: 192.168.10.10
    red_servicios:
      ipv4_address: 192.168.20.10
  depends_on:
    - db
```

```

db:
  image: mysql:5.7
  volumes:
    - ./db_data:/var/lib/mysql
  environment:
    MYSQLROOTPASSWORD: root_password
    MYSQLDATABASE: wordpress
    MYSQLUSER: wordpress
    MYSQLPASSWORD: wordpress
  networks:
    red_servicios:
      ipv4_address: 192.168.20.11

networks:
  red_web:
    driver: bridge
  ipam:
    config:
      - subnet: 192.168.10.0/24
        gateway: 192.168.10.1
  red_servicios:
    driver: bridge
  ipam:
    config:
      - subnet: 192.168.20.0/24
        gateway: 192.168.20.1

```

Explicación del Código

1. Contenedores de WordPress y MySQL:

- `wordpress`: Configura WordPress con variables de entorno para conectarse a la base de datos MySQL.
- `db`: Configura un contenedor MySQL con las credenciales necesarias.

2. Configuración de Volúmenes:

- Los volúmenes permiten que los datos persistan entre reinicios y actualizaciones de contenedores.

3. Redes:

- Ambos contenedores (`wordpress` y `db`) están conectados a `red_servicios` para que WordPress pueda comunicarse con MySQL.
- `wordpress` también está conectado a `red_web` para ser accesible a través del balanceador de carga.

4. Balanceador de carga Nginx:

- Debes actualizar la configuración de Nginx (`nginx.conf`) para incluir las nuevas instancias de WordPress en el balanceo de carga.

Configuración de Nginx

Actualiza tu configuración de Nginx para incluir el nuevo backend (WordPress):

```

# Archivo nginx.conf
http {
    upstream backend {
        server 192.168.10.2;
        server 192.168.10.3;
        server 192.168.10.4;
        server 192.168.10.5;
        server 192.168.10.6;
        server 192.168.10.7;
        server 192.168.10.8;
        server 192.168.10.9;
        server 192.168.10.10; # WordPress
    }

    server {
        listen 80;

```

```

listen 443 ssl;
server_name example.com;

ssl_certificate /etc/nginx/ssl/certificado_MarioRgzLpz.crt;
ssl_certificate_key /etc/nginx/ssl/certificado_MarioRgzLpz.key;

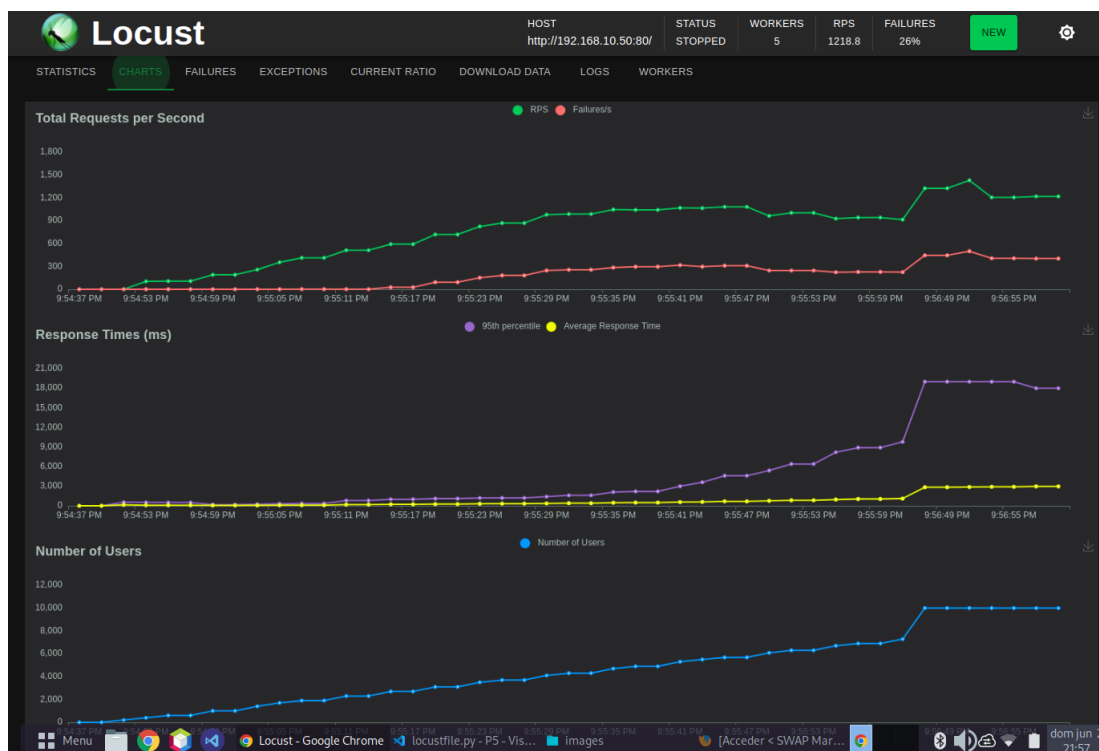
location / {
    proxy_pass http://backend;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Proto $scheme;
}
}


```

Análisis Tras la primera respuesta que no hacia lo que yo quería, aunque si que me ayudaba a configurar correctamente nginx, decido buscar en google y el blog que encuentro hace que me funcione correctamente, comenzando con la instalación que muestro antes.

A3.EJECUCIÓN Y ANÁLISIS DE CARGAS DE PRUEBA AVANZADAS SOBRE CMS

El test a realizar va a ser el mismo que los que he hecho en el apartado B5 para locust, llegar a 10000 usuarios en concurrencia y comprobar entonces las distintas métricas que me muestra locust. Comenzando con http obtengo lo siguiente:





Locust

HOST

http://192.168.10.50:80/

STATUS

STOPPED

WORKERS

5


RPS

1218.8

FAILURES

26%

NEW



STATISTICS

CHARTS

FAILURES

EXCEPTIONS


CURRENT RATIO

DOWNLOAD DATA

LOGS

WORKERS

Type	Name	# Requests	# Fails	Median (ms)	95%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
GET	/index.php	11683	47	510	17000	20000	2842.06	1	21735	87.65	116.4	0.5
POST	/wp-comment-post.php	47195	173	510	17000	20000	2822.28	1	21954	87.68	492.1	1.2
GET	/wp-load.php	23430	11994	820	17000	20000	3049.27	10	22575	1227.62	245.6	160.7
POST	/wp-login.php	35228	18035	890	18000	20000	3165.81	9	24310	4574.28	364.7	241
	Aggregated	117516	30249	660	17000	20000	2972.48	1	24310	1659.91	1218.8	403.4



Locust

HOST

http://192.168.10.50:80/

STATUS

STOPPED

WORKERS

5


RPS

1218.8

FAILURES

26%

NEW



STATISTICS

CHARTS

FAILURES

EXCEPTIONS


CURRENT RATIO

DOWNLOAD DATA

LOGS

WORKERS

# Failures	Method	Name	Message
45	GET	/index.php	RemoteDisconnected("Remote end closed connection without response")
2	GET	/index.php	ConnectionResetError(104, 'Connection reset by peer')
151	POST	/wp-comment-post.php	RemoteDisconnected("Remote end closed connection without response")
22	POST	/wp-comment-post.php	ConnectionResetError(104, 'Connection reset by peer')
6	GET	/wp-load.php	ConnectionResetError(104, 'Connection reset by peer')
63	GET	/wp-load.php	RemoteDisconnected("Remote end closed connection without response")
11925	GET	/wp-load.php	HTTPError("500 Server Error: Internal Server Error for url: /wp-load.php")
16	POST	/wp-login.php	ConnectionResetError(104, 'Connection reset by peer')
102	POST	/wp-login.php	RemoteDisconnected("Remote end closed connection without response")
17917	POST	/wp-login.php	HTTPError("500 Server Error: Internal Server Error for url: /wp-login.php")



Locust

HOST

https://192.168.10.50:443/

STATUS

STOPPING

WORKERS

5

RPS

144.7


FAILURES

1%

EDIT

LOADING

RESET



STATISTICS

CHARTS

FAILURES

EXCEPTIONS

CURRENT RATIO

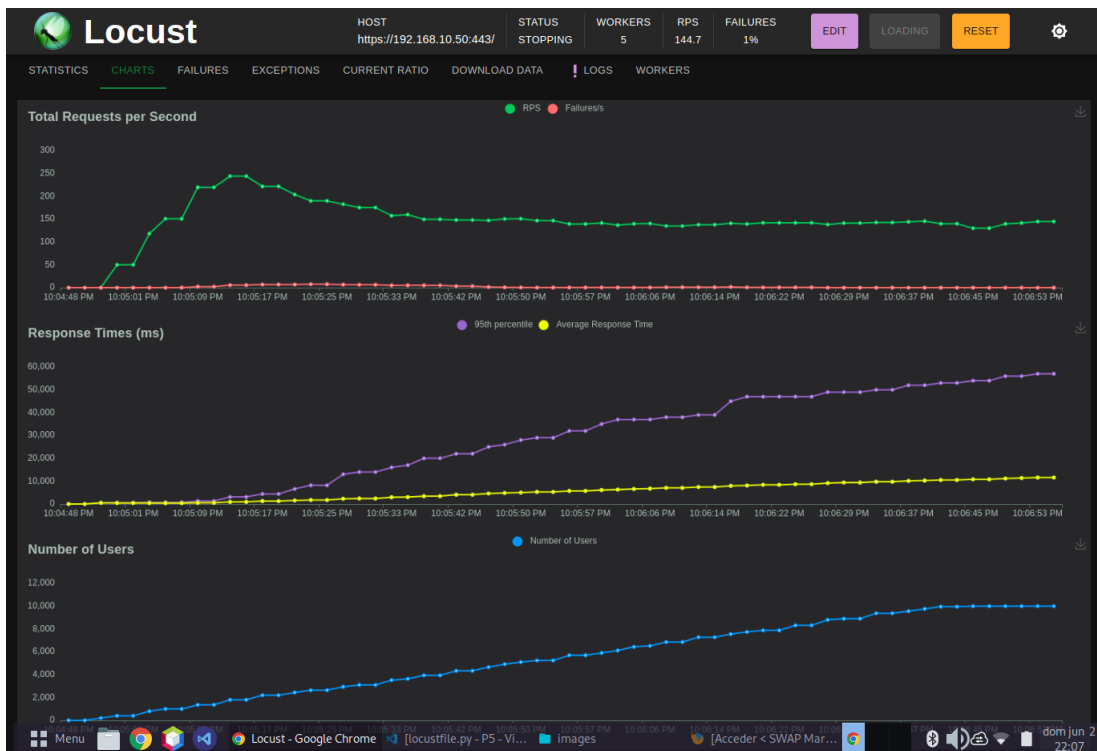
DOWNLOAD DATA

LOGS

WORKERS

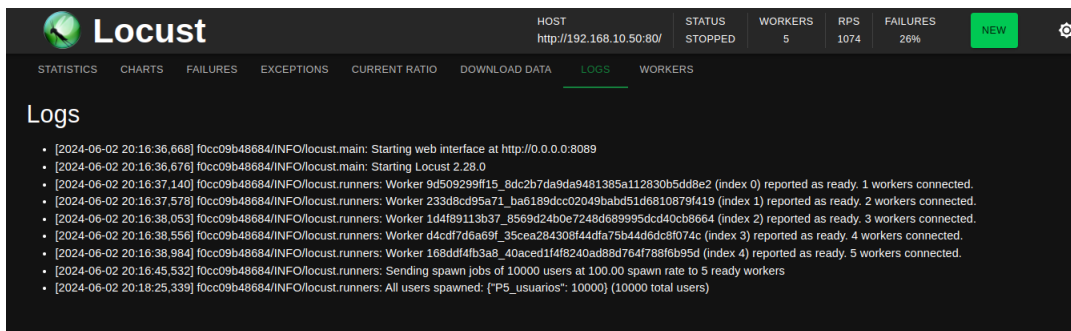
Logs

- [2024-06-02 20:04:44.537] focc09b48684/INFO/locust.main: Starting web interface at http://0.0.0.0:8089
- [2024-06-02 20:04:44.546] focc09b48684/INFO/locust.main: Starting Locust 2.28.0
- [2024-06-02 20:04:44.840] focc09b48684/INFO/locust.runners: Worker 9d509299f15_1f2c48325f9748e5ad64d551168b3ee2 (index 0) reported as ready. 1 workers connected.
- [2024-06-02 20:04:45.156] focc09b48684/INFO/locust.runners: Worker 233d8cd95a71_80a5d57e463047689446b5b11c3039c9 (index 1) reported as ready. 2 workers connected.
- [2024-06-02 20:04:45.309] focc09b48684/INFO/locust.runners: Worker 1d4f89113b37_93771e653f5e402ab394fb10579cd885 (index 2) reported as ready. 3 workers connected.
- [2024-06-02 20:04:45.786] focc09b48684/INFO/locust.runners: Worker d4cdf7d6a69f_30144ee588324563b25426c68569cc6b (index 3) reported as ready. 4 workers connected.
- [2024-06-02 20:04:46.075] focc09b48684/INFO/locust.runners: Worker 168ddf4fb3a8_a81b4502e6c643449dd7665b0fde7673 (index 4) reported as ready. 5 workers connected.
- [2024-06-02 20:04:53.990] focc09b48684/INFO/locust.runners: Sending spawn jobs of 10000 users at 100.00 spawn rate to 5 ready workers
- [2024-06-02 20:05:10.805] focc09b48684/WARNING/locust.runners: Worker 9d509299f15_1f2c48325f9748e5ad64d551168b3ee2 (index 0) exceeded cpu threshold (will only log this once per worker)
- [2024-06-02 20:05:10.923] focc09b48684/WARNING/locust.runners: Worker 233d8cd95a71_80a5d57e463047689446b5b11c3039c9 (index 1) exceeded cpu threshold (will only log this once per worker)
- [2024-06-02 20:05:11.360] focc09b48684/WARNING/locust.runners: Worker 1d4f89113b37_93771e653f5e402ab394fb10579cd885 (index 2) exceeded cpu threshold (will only log this once per worker)
- [2024-06-02 20:05:12.038] focc09b48684/WARNING/locust.runners: Worker 168ddf4fb3a8_a81b4502e6c643449dd7665b0fde7673 (index 4) exceeded cpu threshold (will only log this once per worker)
- [2024-06-02 20:05:12.057] focc09b48684/WARNING/locust.runners: Worker d4cdf7d6a69f_30144ee588324563b25426c68569cc6b (index 3) exceeded cpu threshold (will only log this once per worker)
- [2024-06-02 20:06:34.073] focc09b48684/INFO/locust.runners: Spawning is complete and report waittime is expired, but not all reports received from workers: (*P5_usuarios*: 9380) (9380 total users)

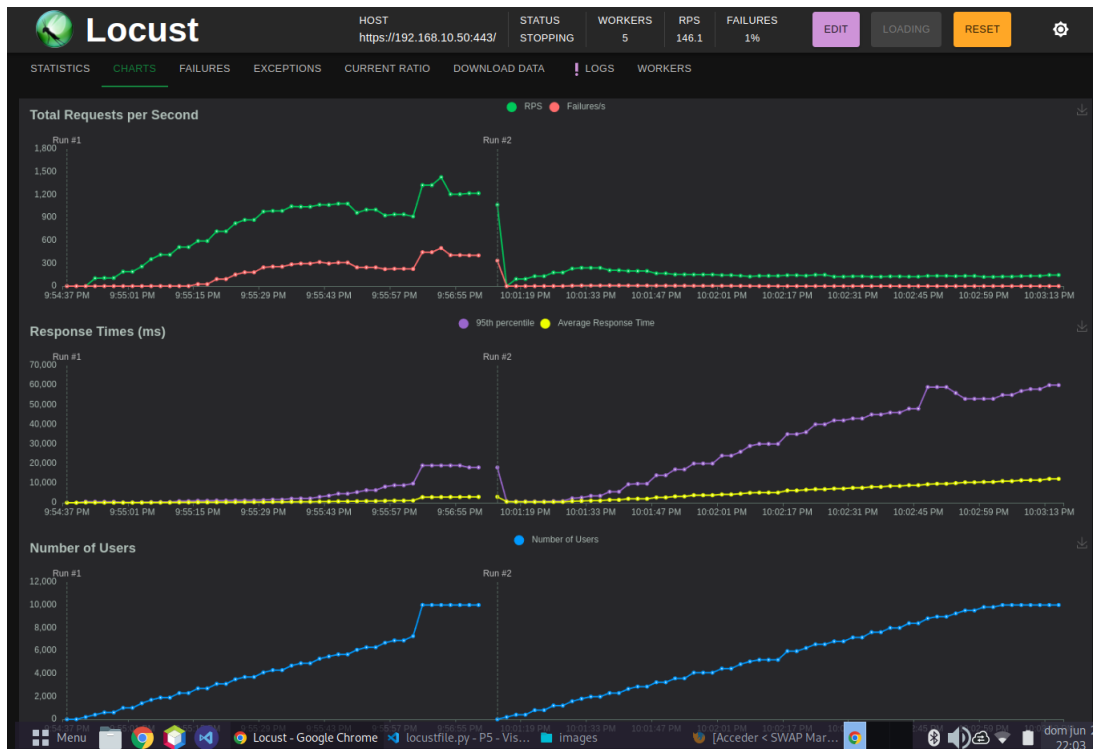


Type	Name	# Requests	# Fails	Median (ms)	95thile (ms)	99thile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
GET	/index.php	1722	22	11000	47000	59000	12400.78	2	90983	86.88	14.4	0
POST	/wp-comment-post.php	7242	81	11000	46000	63000	12064.95	1	94857	87.02	59.3	0
GET	/wp-load.php	3640	45	11000	45000	61000	12001.32	11	94859	0	29.2	0
POST	/wp-login.php	5385	74	11000	46000	63000	12318.38	13	94860	6764.74	43.2	0
Aggregated		17989	222	11000	46000	61000	12160.09	1	94860	2068.37	146.1	0

# Failures	Method	Name	Message
22	GET	/index.php	RemoteDisconnected('Remote end closed connection without response')
75	POST	/wp-comment-post.php	SSLError(8, 'EOF occurred in violation of protocol (_ssl.c:2427)')
6	POST	/wp-comment-post.php	RemoteDisconnected('Remote end closed connection without response')
45	GET	/wp-load.php	RemoteDisconnected('Remote end closed connection without response')
9	POST	/wp-login.php	RemoteDisconnected('Remote end closed connection without response')
65	POST	/wp-login.php	SSLError(8, 'EOF occurred in violation of protocol (_ssl.c:2427)')



Comparando los dos juntos:



Análisis En este caso vamos a comparar los cambios que ha habido con respecto al anterior test de carga en el que solo hacíamos get del index, por lo que no hablaré sobre todas las métricas ni las explicare aquí ya que he hablado en el apartado B5 con anterioridad.

Comenzamos viendo como claramente en http las request por segundo disminuyen en gran cantidad quedando con un pico de unas 1500 mientras que antes teníamos un pico de 3000. Esto provoca que el numero de request totales baje también viendo como en total tenemos 117516 que es menos de la mitad que con el test anterior. Para https el cambio no es tan grande viendo como el pico máximo es de casi 250 y el total de peticiones casi 18000 suponiendo una disminución de unas 5000 peticiones. Esto debido al uso de TLS 1.3 y las cookies haciendo que el número de peticiones sea más estable que en http a pesar del aumento de carga.

El tiempo de respuesta también ha aumentado en gran medida en http pasando a tener un promedio de 3000 ms siendo esto un aumento de unas 70 veces más (antes era de 44ms) mientras que el tiempo en https ha aumentado en unos 1000ms siendo esto un 10 % aproximadamente.

También podemos ver el gran aumento de fallos en http en el que vemos el error `HTTPError('500 Server Error')` que nos dice que el servidor encontro una condición inesperada que no le permite solucionarlo (extraido de [aqui](#)). Esto no ocurre en https por lo que el numero de fallos es mucho menor, manteniendo un numero similar que en el anterior test posiblemente porque los workers llegan al uso máximo de CPU. En los logs podemos ver como en http al recibir muchas menos peticiones los workers no llegan al máximo de CPU pero aun así el número de errores ha aumentado. En https sigue apareciendo el mensaje indicando que se llega al nivel de uso máximo de CPU.

Comparando las dos podemos ver que el más perjudicado con respecto al primer test ha sido http que ha visto reducido a la mitad el número de peticiones y aumentado en gran medida los fallos.