Memoria - Monitorización APIWEB

Autor: Lucas Hidalgo Herrera
Asignatura: Ingeniería de Servidores
Grado: Doble Grado en Ingeniería Informática y Matemáticas

1. Introducción

Tal y como aparece en el guión de prácticas, en esta memoria sobre la monitorización de la API WEB de la Escuela Técnica Superior de Ingeniería Informática y Telecomunicaciones, aparecerá el resultado de realizar un *Dashboard* de *Grafana* creado para dicha monitorización así como los resultados obtenidos.

2. Configuración de Prometheus

Antes de nada, debemos configurar el contenedor de *Prometheus* para que pueda tomar las métricas expuestas por *NodeJS* en la API Web. Primero, ponemos en funcionamiento la API y nuestro contenedor de *Grafana+Prometheus* con la configuración de partida, que será la configuración final del ejercicio anterior.

Prometheus+Grafana:

```
[lucas@lhhhost:20:26:10 progra]$ docker compose up
WARN[0000] /home/el_dramas/Desktop/Infomates/Tercer_Curso/Segundo_Cuatri/ISE/Pra
cticas/Bloque_2/Monitoring/progra/docker-compose.yml: the attribute `version` is
 obsolete, it will be ignored, please remove it to avoid potential confusion
WARN[0000] Found orphan containers ([progra-systemd-exporter-1 progra-node_expor
ter-1]) for this project. If you removed or renamed this service in your compose
 file, you can run this command with the --remove-orphans flag to clean it up.
[+] Running 2/2
 ✔ Container progra-grafana-1     Created                                   0.0s
 ✔ Container progra-prometheus-1  Recre...                                  0.1s
Attaching to grafana-1, prometheus-1
prometheus-1  | ts=2025-05-19T18:35:14.144Z caller=main.go:564 level=info msg="N
o time or size retention was set so using the default time retention" duration=1
5d
prometheus-1  | ts=2025-05-19T18:35:14.145Z caller=main.go:608 level=info msg="S
tarting Prometheus Server" mode=server version="(version=2.50.0, branch=HEAD, re
```

API Web:

```
[lucas@lhhhost:20:37:00 Obligatorio_ii]$ cd iseP4JMeter/
[lucas@lhhhost:20:37:03 iseP4JMeter]$ ls
docker-compose.yml  jMeter      mongodb  pruebaEntorno.sh  results.jtl
images              jmeter.log  nodejs   README.md
[lucas@lhhhost:20:37:03 iseP4JMeter]$ docker compose up
WARN[0000] /home/el_dramas/Desktop/Infomates/Tercer_Curso/Segundo_Cuatri/ISE/Pra
cticas/Bloque_2/Obligatorio_ii/iseP4JMeter/docker-compose.yml: the attribute `ve
rsion` is obsolete, it will be ignored, please remove it to avoid potential conf
usion
[+] Running 3/3
 ✔ Container isep4jmeter-mongodb-1      Created                               0.0s
 ✔ Container isep4jmeter-mongodbinit-1  Created                               0.0s
 ✔ Container isep4jmeter-nodejs-1       Created                               0.0s
Attaching to mongodb-1, mongodbinit-1, nodejs-1
mongodb-1      | {"t":{"$date":"2025-05-19T18:37:09.872+00:00"},"s":"I",  "c":"C
ONTROL",  "id":23285,   "ctx":"main","msg":"Automatically disabling TLS 1.0, to
force-enable TLS 1.0 specify --sslDisabledProtocols 'none'"}
mongodb-1      | {"t":{"$date":"2025-05-19T18:37:09.874+00:00"},"s":"I",  "c":"N
ETWORK",  "id":4915701, "ctx":"main","msg":"Initialized wire specification","att
r":{"spec":{"incomingExternalClient":{"minWireVersion":0,"maxWireVersion":17},"i
ncomingInternalClient":{"minWireVersion":0,"maxWireVersion":17},"outgoing":{"min
WireVersion":6,"maxWireVersion":17},"isInternalClient":true}}}
```

No obstante, es claro que esto no funciona pues *Prometheus* desconoce el *endPoint* de la API Web; por tanto, usaremos la opción de *extra hosts* en el *docker_compose.yml* para "comunicarle" a *Prometheus* un nuevo *endPoint*.

De la misma manera, crearemos una nueva tarea en el archivo *prometheus.yml* que asigne la dirección IP correcta (*host.docker.internal*) y el puerto *3000*, el puerto de la API.

docker compose.yml:

```yaml
---
version: "3"
services:
  prometheus:
    image: prom/prometheus:v2.50.0
    extra_hosts:
      - "host.docker.internal:host-gateway"
    ports:
      - 9090:9090
    volumes:
      - ./prometheus_data:/prometheus
      - ./prometheus.yml:/etc/prometheus/prometheus.yml
    command:
      - "--config.file=/etc/prometheus/prometheus.yml"
  grafana:
    image: grafana/grafana:9.1.0
    ports:
      - 4000:3000
    volumes:
      - ./grafana_data:/var/lib/grafana
    depends_on:
      - prometheus
```

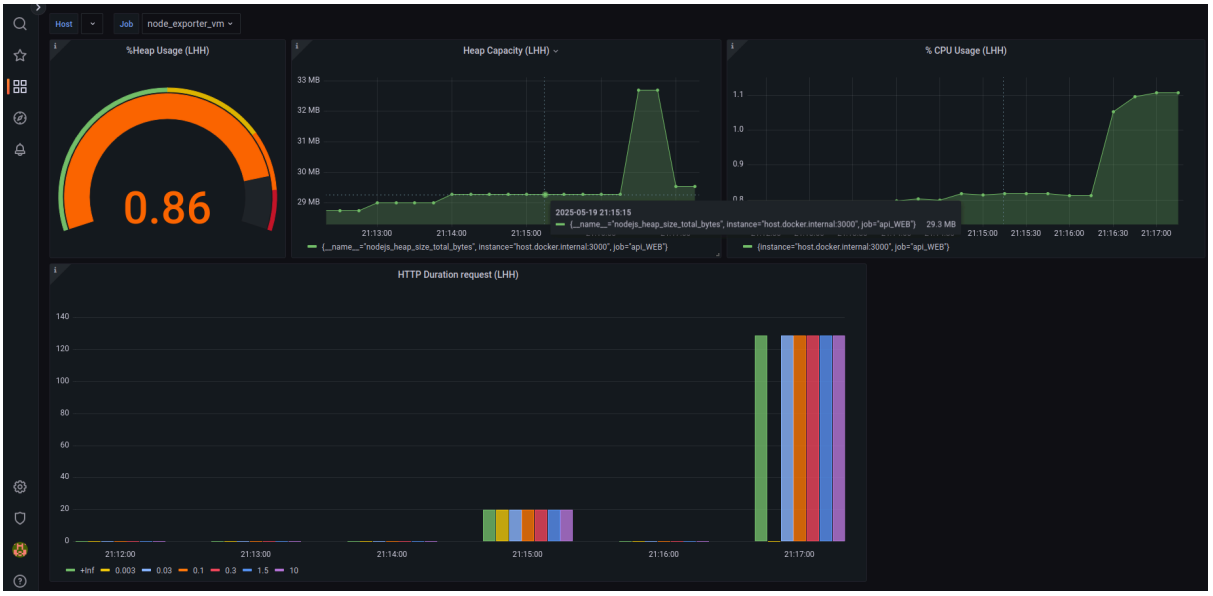
prometheus.yml:

```yaml
global:
  scrape_interval: 5s
scrape_configs:
  - job_name: "prometheus_service"
    static_configs:
      - targets: ["prometheus:9090"]

  - job_name: "node_exporter_vm"
    static_configs:
      - targets: ["192.168.57.3:9100"]

  - job_name: "api_WEB"
    static_configs:
      - targets: ["host.docker.internal:3000"]
```

Veamos ahora que todo funciona en orden mostrando ya el *Dashboard*(*Monitoring the api*) que he creado para monitorizar los aspectos que se piden en el ejercicio:

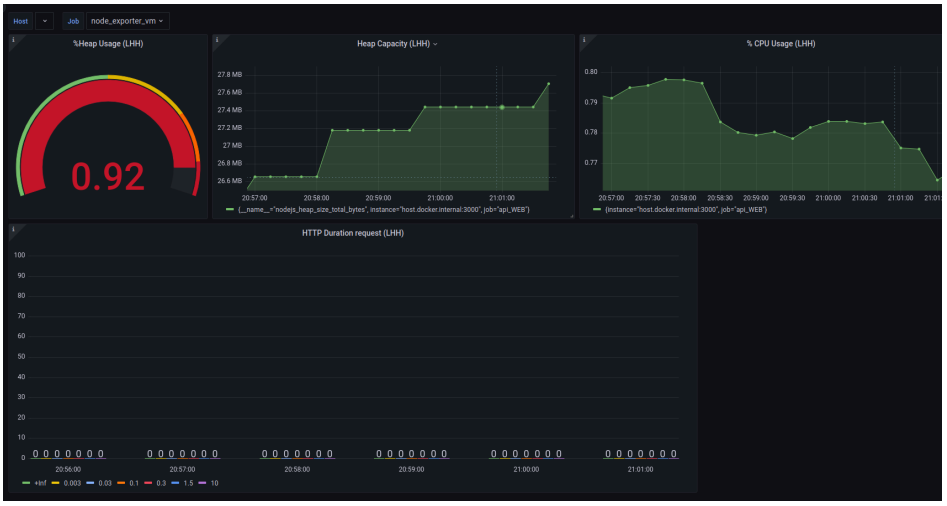


En esta imagen, aparecen las cuatro métricas que nos piden:

- **Tiempos de respuesta de los endPoints:** he decidido realizar un gráfico de barras que recoja, para cada minuto, el número de peticiones resueltas por debajo de cada una de los siguientes tiempos y en este orden: +Inf, 0.003s, 0.03s, 0.1s, 0.3s, 1.5s 10s.
- **Porcentaje de Heap:** he decidido usar un gráfico tipo *Gauge* que represente dicho porcentaje siendo posible la monitorización constante.
- **Porcentaje de uso de CPU:** es una serie temporal al igual que se realizó en la monitorización de la máquina virtual con el sistema operativo *Rocky Linux*.
- **Capacidad del heap:** simplemente se ha utilizado la métrica expuesta en el guión y se ha representado sobre una serie temporal.

Vamos a ejecutar varias veces la prueba de carga de *JMeter* realizada para el segundo ejercicio a entregar; en mi caso se llama *Obligatorio.jmx*.

Antes:



Ejecución de la prueba de carga:

```
[lucas@lhhhost:21:02:58 jMeter]$ ~/apache-jmeter-5.6.3/bin/jmeter -n -t Obligatorio_ii.jmx
-l results.jtl
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and
will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and
will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and
will be removed in a future release
WARN StatusConsoleListener The use of package scanning to locate plugins is deprecated and
will be removed in a future release
Creating summariser <summary>
Created the tree successfully using Obligatorio_ii.jmx
Starting standalone test @ 2025 May 19 21:03:07 CEST (1747681387875)
Waiting for possible Shutdown/StopTestNow/HeapDump/ThreadDump message on port 4445
Warning: Nashorn engine is planned to be removed from a future JDK release
summary =    118 in 00:00:11 =   10.7/s Avg:    14 Min:     6 Max:    96 Err:     0 (0.00%)
Tidying up ...    @ 2025 May 19 21:03:19 CEST (1747681399581)
... end of run
```

Después:

Como podemos apreciar en la imagen, se han recogido el número de peticiones resueltas en la hora 21:04:00, que ha sido de **129** y se han resuelto todas antes de los **0.03s**. Además entre los segundos 21:03:00 y 21:03:30 se denota un aumento de la carga de uso de la CPU a casi el **100%**.

Pasado un tiempo se puede ver que los niveles vuelven a estabilizarse.