

# Objetivo

Aplicar buenas prácticas de rendimiento, seguridad y mantenimiento en los contenedores Docker utilizados en el entorno de monitorización con Prometheus, cAdvisor y Grafana

## Tareas realizadas

### 1. Limitación de recursos por contenedor

Se añadieron límites explícitos de **CPU** y **memoria RAM** para evitar el consumo excesivo de recursos por parte de los servicios. Esto se aplicó a los tres contenedores principales (prometheus, cadvisor, grafana) mediante la directiva `deploy.resources.limits`.

### 2. Uso de versiones específicas y ligeras de imágenes

Para evitar problemas por cambios en versiones latest y asegurar mayor estabilidad, se fijaron versiones concretas de cada imagen Docker:

- prom/prometheus:v2.52.0
- gcr.io/cadvisor/cadvisor:v0.49.1
- grafana/grafana:10.3.3

Además, se podrían considerar imágenes alpine en futuros servicios si se busca mayor reducción de tamaño.

### 3. Variables de entorno con archivo .env

Se eliminaron los puertos hardcodeados y se sustituyeron por variables configurables en un archivo `.env`. Esto permite una mayor reutilización del `docker-compose.yml` en distintos entornos sin necesidad de modificar el archivo principal.

#### 4. Reducción de puertos expuestos

Solo se expusieron los puertos necesarios para acceder a los servicios desde el host:

- Prometheus: 9090
- Grafana: 3000
- cAdvisor: 8081 (puede restringirse a localhost en producción)

No se dejaron puertos abiertos innecesariamente.

#### 5. Red en bridge aislada

Todos los servicios fueron conectados a una red llamada **monitoring** para aislar la comunicación entre ellos y no usar la red por defecto.

#### 6. Persistencia de datos

Se mantuvo la persistencia de los datos mediante volúmenes nombrados, lo que asegura que la información de Prometheus y Grafana no se pierda al reiniciar los contenedores:

### Resultado Final

Un ***docker-compose.yml*** mucho más robusto, reutilizable, eficiente y seguro, ideal para entornos de desarrollo o incluso pruebas previas a producción.



daniel > Sprint1 > Prometheus > 🔥 docker-compose.yml > ...


docker-compose.yml - The Compose specification establishes a standard for the definition of multi-container platform-ag

```

1  version: "3"
2
3  >Run All Services
4  services:
5    >Run Service
6    prometheus:
7      image: prom/prometheus:v2.52.0
8      ports:
9        - "${PROMETHEUS_PORT}:9090"
10     volumes:
11       - ./prometheus.yml:/etc/prometheus/prometheus.yml
12       - prometheus_data:/prometheus
13     networks:
14       - monitoring
15     deploy:
16       resources:
17         limits:
18           cpus: "0.5"
19           memory: "512M"
20
21   >Run Service
22   cadvisor:
23     image: gcr.io/cadvisor/cadvisor:v0.49.1
24     ports:
25       - "${CADVISOR_PORT}:8081"
26     volumes:
27       - /:/rootfs:ro
28       - /var/run:/var/run:ro
29       - /sys:/sys:ro
30       - /var/lib/docker:/var/lib/docker:ro
31     networks:
32       - monitoring
33     deploy:
34       resources:
35         limits:
36           cpus: "0.5"
37           memory: "512M"
38
39   >Run Service
40   grafana:
41     image: grafana/grafana:10.3.3
42     ports:
43       - "${GRAFANA_PORT}:8081"
44     volumes:
45       - grafana_data:/var/lib/grafana
46     networks:
47       - monitoring
48     deploy:
49       resources:
50         limits:
51           cpus: "0.5"
52           memory: "512M"
53
54   volumes:
55     prometheus_data:
56     grafana_data:
57   networks:
58     monitoring:
59       driver: bridge

```

 `.env` 

daniel > Sprint1 > Prometheus >  `.env`

1

`PROMETHEUS_PORT=9090`

2

`GRAFANA_PORT=3000`

3

`CADVISOR_PORT=8081`

4