8. Optimización de Configuraciones

Para mejorar el rendimiento, la organización y la seguridad de los contenedores y servicios usando buenas prácticas,.

EJERCICIO 3

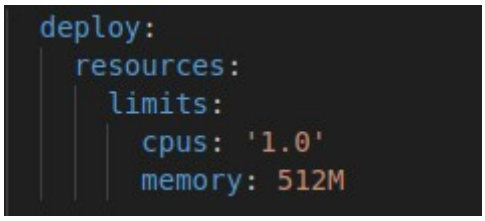
Vamos a modificar el archivo docker-compose.yml:

Sprint-1 > Ejercicio-3 > docker-compose.yml

```yaml
Run All Services
services:
  Run Service
  nginx:
    deploy:
      resources:
        limits:
          cpus: '1.0'
          memory: 512M

    image: nginx:alpine
    container_name: nginx
    ports:
      - "8080:80"
    depends_on:
      - php
    networks:
      - network

  Run Service
  php:
    deploy:
      resources:
        limits:
          cpus: '0.50'
          memory: 512M
    image: php:8.1-apache
    container_name: php
    networks:
      - network

  Run Service
  mysql:
    deploy:
      resources:
        limits:
          cpus: '0.50'
          memory: 512M
    image: mysql:8.0
    container_name: mysql
    volumes:
      - mysql-data:/var/lib/mysql
    environment:
      - MYSQL_ROOT_PASSWORD=${MYSQL_ROOT_PASSWORD}
      - MYSQL_DATABASE=${MYSQL_DATABASE}
      - MYSQL_USER=${MYSQL_USER}
      - MYSQL_PASSWORD=${MYSQL_PASSWORD}
    networks:
      - network
```

1. Limitar recursos de CPU y memoria de los contenedores

Para mejorar el rendimiento y evitar que los contenedores usen más recursos de los necesarios, añadimos límites de CPU y memoria con:



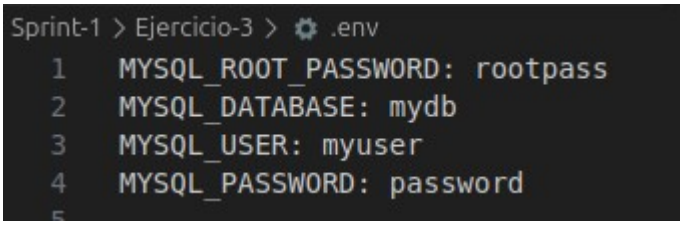
Este bloque se añadirá a los contenedores nginx, php y mysql.

2. Utilizar imagen ligera alpine

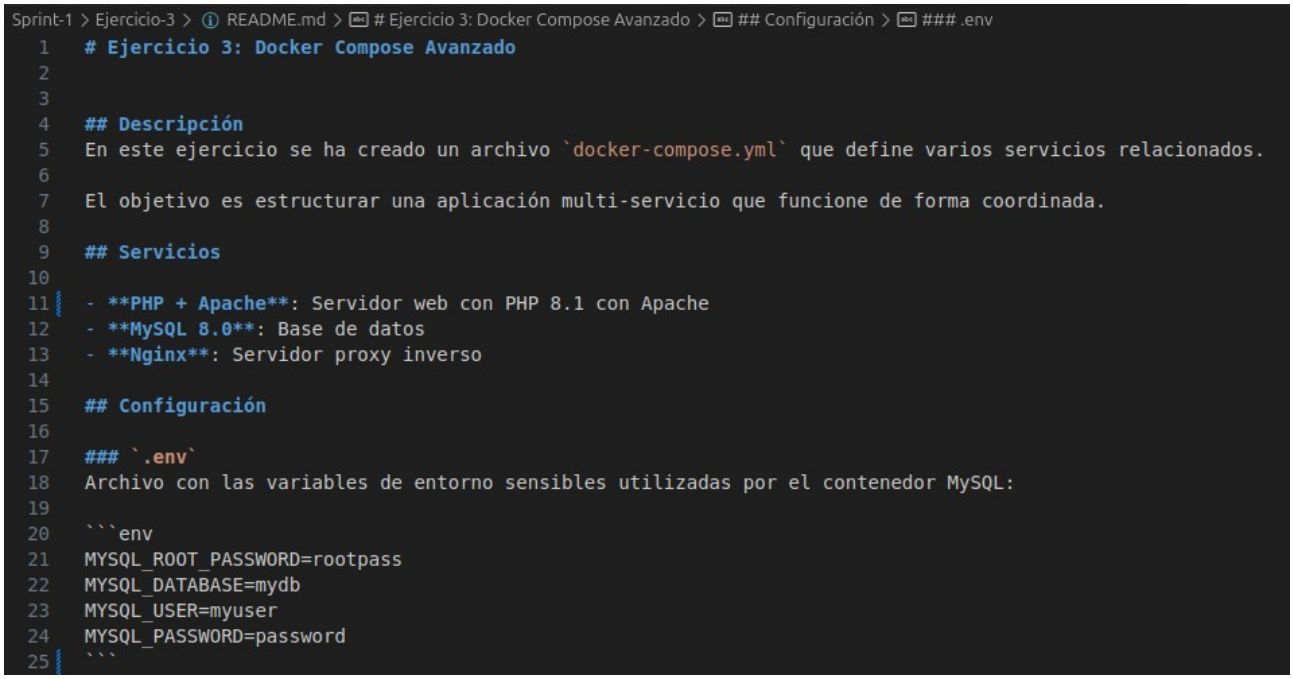
Usamos la imagen alpine en el contenedor nginx para reducir el tamaño del contenedor y mejorar su rendimiento.

3. Crear archivo .env

Creamos un archivo .env para el contenedor mysql dónde definiremos las variables de entornos sensibles, y evitamos no exponerlas en el archivo docker-compose.yml.



4. Vamos a crear un archivo README.md para documentar los servicios y configuraciones



EJERCICIO 4

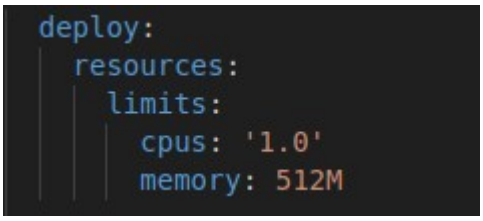
Vamos a modificar el archivo docker-compose.yml:

Sprint-1 > Ejercicio-4 > docker-compose.yml

```yaml
version: "3.8"
Run All Services
services:
  Run Service
  nginx:
    deploy:
      resources:
        limits:
          cpus: '1.0'
          memory: 512M
    image: nginx:alpine
    container_name: nginx
    ports:
      - "8080:80"
    volumes:
      - ./www:/var/www/html
      - ./nginx/default.conf:/etc/nginx/conf.d/default.conf
    depends_on:
      - php
    networks:
      - network

  Run Service
  php:
    deploy:
      resources:
        limits:
          cpus: '1.0'
          memory: 512M
    image: php:8.1-fpm-alpine
    container_name: php
    volumes:
      - ./www:/var/www/html
    networks:
      - network

networks:
  network:
    name: network
```

1. Limitar recursos de CPU y memoria de los contenedores

Para mejorar el rendimiento y evitar que los contenedores usen más recursos de los necesarios, añadimos límites de CPU y memoria con:

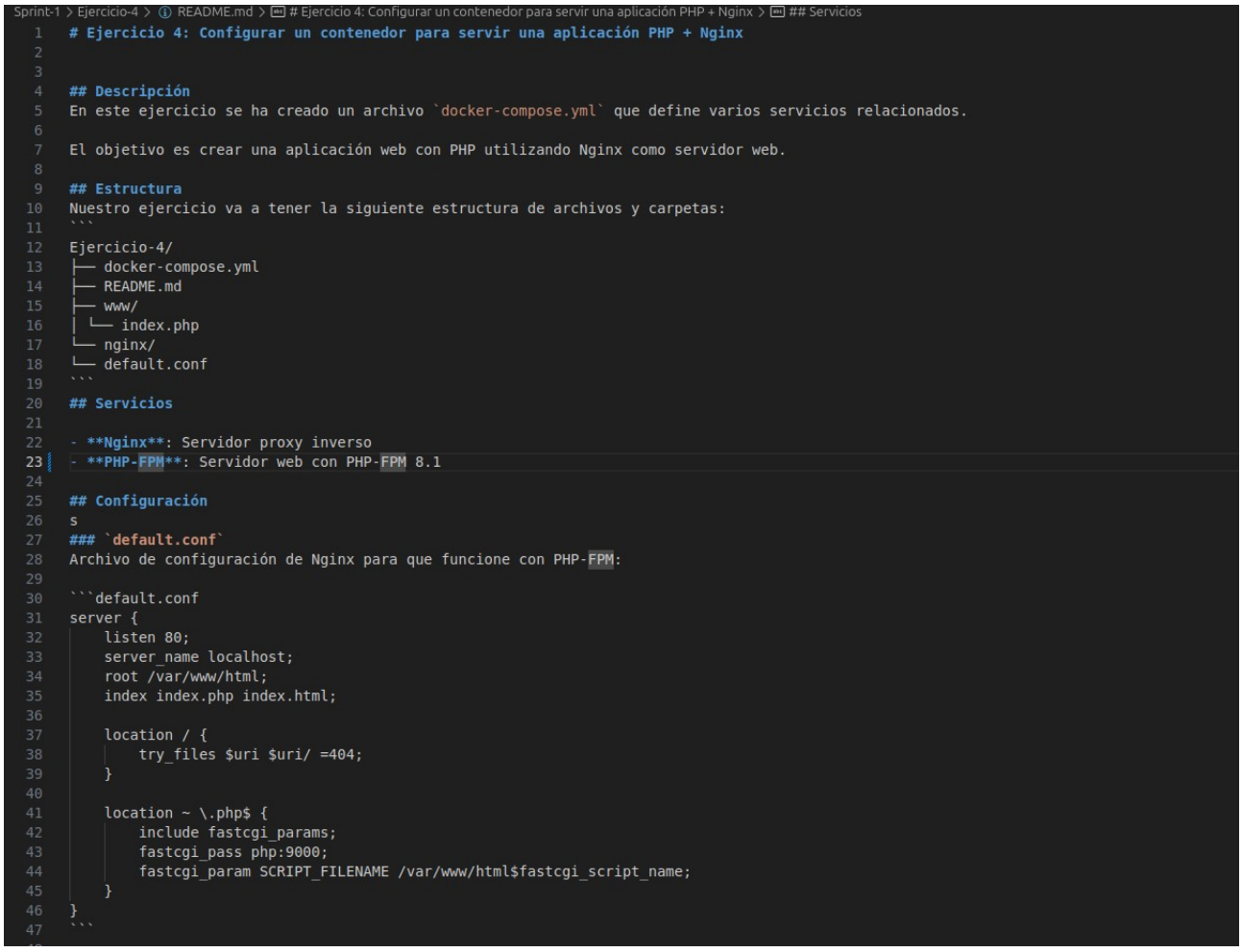


Este bloque se añadirá a los contenedores nginx y php.

2. Utilizar imagen ligera alpine

Usamos la imagen alpine en los contenedores nginx y php para reducir el tamaño del contenedor y mejorar su rendimiento.

3. Creamos un archivo README.md para documentar los servicios y configuraciones



````
### `index.php`
Archivo de prueba para verificar que Nginx y PHP-FPM están funcionando de manera correcta:

```index.php
<?php
phpinfo();
?>
```
````

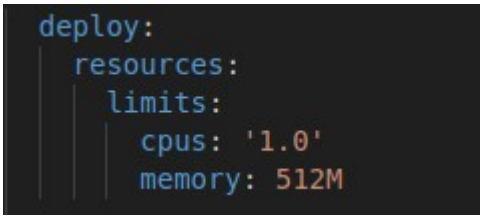
EJERCICIO 6

Vamos a modificar el archivo docker-compose.yml:

```yaml
version: "3.8"
services:
  prometheus:
    deploy:
      resources:
        limits:
          cpus: '1.0'
          memory: 512M
    image: prom/prometheus:latest
    container_name: prometheus
    ports:
      - "9090:9090"
    volumes:
      - ./prometheus.yml:/etc/prometheus/prometheus.yml
    networks:
      - monitoring

  cadvisor:
    deploy:
      resources:
        limits:
          cpus: '1.0'
          memory: 512M
    image: gcr.io/cadvisor/cadvisor:latest
    container_name: cadvisor
    ports:
      - "8080:8080"
    volumes:
      - /:/rootfs:ro
      - /var/run:/var/run:rw
      - /sys:/sys:ro
      - /dev/disk/:/dev/disk:ro
    networks:
      - monitoring

networks:
  monitoring:
    driver: bridge
```

1. Limitar recursos de CPU y memoria de los contenedores

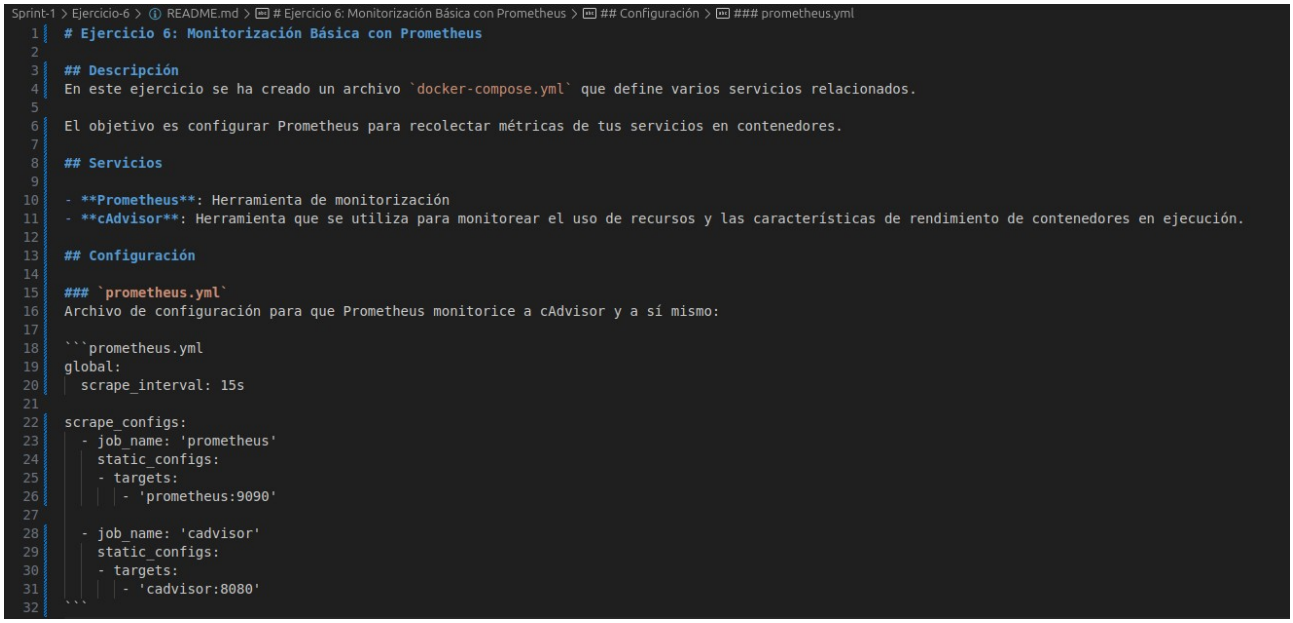
Para mejorar el rendimiento y evitar que los contenedores usen más recursos de los necesarios, añadimos límites de CPU y memoria con:



```yaml
deploy:
  resources:
    limits:
      cpus: '1.0'
      memory: 512M
```

Este bloque se añadirá a los contenedores prometheus y cadvisor.

2. Creamos un archivo README.md para documentar los servicios y configuraciones



Sprint-1 > Ejercicio-6 > README.md > # Ejercicio 6: Monitorización Básica con Prometheus > ## Configuración > ### prometheus.yml

```
# Ejercicio 6: Monitorización Básica con Prometheus

## Descripción
En este ejercicio se ha creado un archivo `docker-compose.yml` que define varios servicios relacionados.

El objetivo es configurar Prometheus para recolectar métricas de tus servicios en contenedores.

## Servicios

- **Prometheus**: Herramienta de monitorización
- **cAdvisor**: Herramienta que se utiliza para monitorear el uso de recursos y las características de rendimiento de contenedores en ejecución.

## Configuración

### `prometheus.yml`
Archivo de configuración para que Prometheus monitorice a cAdvisor y a sí mismo:

```prometheus.yml
global:
  scrape_interval: 15s

scrape_configs:
  - job_name: 'prometheus'
    static_configs:
    - targets:
        - 'prometheus:9090'

  - job_name: 'cadvisor'
    static_configs:
    - targets:
        - 'cadvisor:8080'
```
```

EJERCICIO 7

Vamos a modificar el archivo docker-compose.yml:

Sprint-1 > Ejercicio-7 > docker-compose.yml

```yaml
version: "3.8"
Run All Services
services:
  Run Service
  prometheus:
    deploy:
      resources:
        limits:
          cpus: '1.0'
          memory: 512M
    image: prom/prometheus:latest
    container_name: prometheus
    ports:
      - "9090:9090"
    volumes:
      - ./prometheus.yml:/etc/prometheus/prometheus.yml
    networks:
      - monitoring

  Run Service
  cadvisor:
    deploy:
      resources:
        limits:
          cpus: '1.0'
          memory: 512M
    image: gcr.io/cadvisor/cadvisor:latest
    container_name: cadvisor
    ports:
      - "8080:8080"
    volumes:
      - /:/roots:ro
      - /var/run:/var/run:rw
      - /sys:/sys:ro
      - ./dev/disk/:/dev/disk:ro
    networks:
      - monitoring

  Run Service
  grafana:
    deploy:
      resources:
        limits:
          cpus: '1.0'
          memory: 512M
    image: grafana/grafana:latest
    container_name: grafana
    ports:
      - "3000:3000"
    volumes:
      - grafana_data:/var/lib/grafana
    networks:
      - monitoring

volumes:
  grafana_data:

networks:
  monitoring:
    driver: bridge
```

1. Limitar recursos de CPU y memoria de los contenedores

Para mejorar el rendimiento y evitar que los contenedores usen más recursos de los necesarios, añadimos límites de CPU y memoria con:

```yaml
deploy:
  resources:
    limits:
      cpus: '1.0'
      memory: 512M
```

Este bloque se añadirá a los contenedores prometheus, cadvisor y grafana.

2. Creamos un archivo README.md para documentar los servicios y configuraciones

Sprint-1 > Ejercicio-7 > README.md > # Ejercicio 7: Instalación de Grafana > ## Configuración > ### prometheus.yml

```markdown
# Ejercicio 7: Instalación de Grafana

## Descripción
En este ejercicio se ha creado un archivo `docker-compose.yml` que define varios servicios relacionados.

El objetivo es configurar Grafana para construir una vista visual del rendimiento de tu entorno en tiempo real.

## Servicios

- **Prometheus**: Herramienta de monitorización
- **cAdvisor**: Herramienta que se utiliza para monitorear el uso de recursos y las características de rendimiento de contenedores en ejecución.
- **Grafana**: Plataforma que se utiliza para la visualización de datos, análisis y monitoreo de infraestructuras de TI.

## Configuración

### `prometheus.yml`
Archivo de configuración para que Prometheus monitorice a cAdvisor y a sí mismo:

```prometheus.yml
global:
  scrape_interval: 15s

scrape_configs:
  - job_name: 'prometheus'
    static_configs:
    - targets:
      - 'prometheus:9090'

  - job_name: 'cadvisor'
    static_configs:
    - targets:
      - 'cadvisor:8080'
```
```