6. Monitorización Básica con Prometheus

Para levantar un contenedor de Prometheus usando Docker Compose, vamos a realizar los siguientes pasos:

1. Creamos nuestro archivo docker-compose.yml en la versión 3.8

Sprint-1 > Ejercicio-6 > docker-compose.yml

```yaml
version: "3.8"
Run All Services
services:
  Run Service
  prometheus:
    image: prom/prometheus:latest
    container_name: prometheus
    ports:
      - "9090:9090"
    volumes:
      - ./prometheus.yml:/etc/prometheus/prometheus.yml
    networks:
      - monitoring

  Run Service
  cadvisor:
    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
```

Vamos a ir explicando paso a paso el código del archivo:

- Vamos a ejecutar 2 servicios: prometheus y cadvisor.
  - Prometheus:
  → Usa la imagen más reciente de prometheus.
  → El nombre del contenedor es prometheus.
  → Se expone el puerto 9090 del contenedor al puerto 9090 de la máquina local.
  → Usa un volumen *./prometheus.yml:/etc/prometheus/prometheus.yml*: monta el archivo prometheus.yml en la ruta de configuración prometheus dentro del contenedor.
  → Está conectada a una red llamada monitoring que permite que todos los contenedores conectados a esa misma red se comuniquen entre sí.

- **cAdvisor:**
  → Usa la imagen Docker de cAdvisor.
  → El nombre del contenedor es cadvisor.
  → Se expone el puerto 8080 del contenedor al puerto 8080 de la máquina local.
  → Usa varios volúmenes del sistema en modo solo lectura para que cAdvisor pueda acceder a información CPU, red, etc.
  → Está conectada a una red llamada monitoring que permite que todos los contenedores conectados a esa misma red se comuniquen entre sí.

- Definimos la red con el nombre monitoring, dónde se conectarán los contenedores entre sí mediante un puente (bridge).

2. Creamos el archivo prometheus.yml

Sprint-1 > Ejercicio-6 > ! prometheus.yml

```yaml
global:
  scrape_interval: 15s

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

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

Vamos a ir explicando paso a paso el código del archivo:

- En global (ajustes generales) se define:
  - scrape_interval: 15 → Establece que prometheus recolecte métricas cada 15 segundos de los servicios definidos.
- En scrape_configs definimos 2 servicios de los cuales prometheus obtendrá los datos:
  - **Servicio prometheus:** Prometheus va a recoger métricas de sí mismo a través del puerto 9090.
  - **Servicio cadvisor:** Prometheus va a recoger métricas del contenedor cadvisor en el puerto 8080.

3. Comprobamos que los servicios estén funcionando

Tras crear todos los archivos, vamos a comprobar que los servicios se puedan lanzar con el comando:

```
docker compose up -d
```

```
anabel@ubuntu:~/Sprint-1/Ejercicio-6$ docker compose up -d
WARN[0000] /home/anabel/Sprint-1/Ejercicio-6/docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion
[+] Running 3/3
 ✔ Network ejercicio-6_monitoring  Created                                                                    0.4s
 ✔ Container prometheus            Started                                                                    1.1s
 ✔ Container cadvisor              Started                                                                    1.1s
anabel@ubuntu:~/Sprint-1/Ejercicio-6$
```

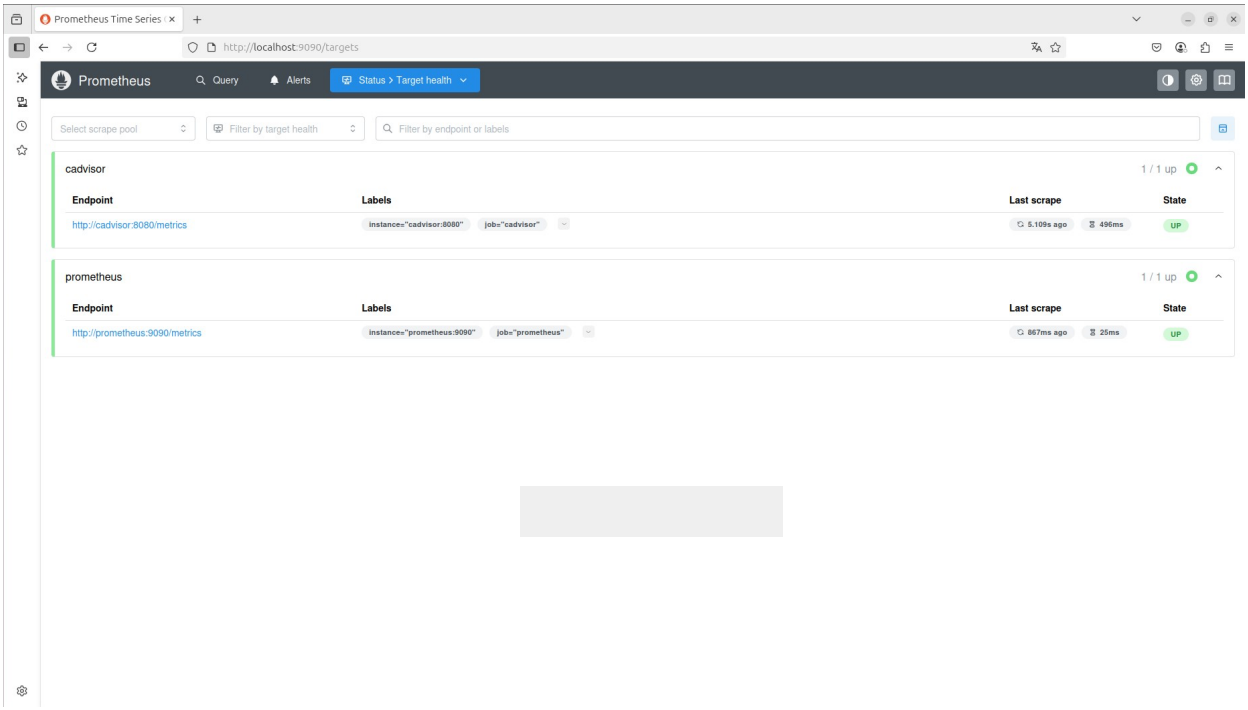
Tras lanzar los servicios, vamos a verificar que los contenedores estén en ejecución con el siguiente comando:

docker compose ps

```
anabel@ubuntu:~/Sprint-1/Ejercicio-6$ docker ps
CONTAINER ID   IMAGE                          COMMAND                  CREATED          STATUS                    PORTS                                                NAMES
6ca3c72e96ef   prom/prometheus:latest         "/bin/prometheus --c…"   34 seconds ago   Up 34 seconds             0.0.0.0:9090->9090/tcp, :::9090->9090/tcp            prometheus
e269c61adb9b   gcr.io/cadvisor/cadvisor:latest "/usr/bin/cadvisor -…"   34 seconds ago   Up 34 seconds (healthy)   0.0.0.0:8080->8080/tcp, :::8080->8080/tcp            cadvisor
anabel@ubuntu:~/Sprint-1/Ejercicio-6$
```

4. Comprobamos que Prometheus funciona

Finalmente, abrimos la dirección http://localhost:9090 desde el navegador. En *Status*, abrimos *Target Health*.



Podemos ver que Prometheus funciona de manera correcta, y al ingresar en los targets nos sale en estado activo (UP) tanto el servicio cadvisor como prometheus.