




PRÁCTICA INTEGRAL: DESPLIEGUE Y OBSERVABILIDAD DE MICROSERVICIOS



Cristóbal Suárez Abad
OPTATIVA - 2º ASIR

Contenido

Práctica Integral: Despliegue y Observabilidad de Microservicios	2
1. Introducción y Objetivos	2
2. Fase 1: Aprovisionamiento de la Aplicación	2
3. Fase 2: El Stack de Monitorización.....	6

Práctica Integral: Despliegue y Observabilidad de Microservicios

1. Introducción y Objetivos

En el mundo real, un administrador de sistemas no solo "instala" servicios, sino que debe asegurar su **disponibilidad** y **rendimiento**. En esta práctica, realizarás el aprovisionamiento de una aplicación multi-contenedor y configurarás un stack profesional de monitorización basado en la arquitectura **Prometheus + Grafana**.

Objetivos técnicos:

- Desplegar una arquitectura de microservicios mediante **Infraestructura como Código (Docker Compose)**.
- Entender la arquitectura de **pull-metrics** de Prometheus.
- Implementar **Exporters** para obtener datos del sistema y de los contenedores.

2. Fase 1: Aprovisionamiento de la Aplicación

Como no tenemos una app previa, tu primera tarea es desplegar la infraestructura base.

1. **El Escenario:** Crea un archivo `docker-compose.yml` que levante los siguientes servicios:
 - **Frontend:** Una imagen de `nginx:alpine`.
 - **Backend:** Una imagen de ejemplo (puedes usar `katacoda/docker-http-server` o una app sencilla en Python).
 - **Base de Datos:** Un contenedor de `redis:alpine`.
2. **Configuración de Red:** Todos los servicios deben estar en una red interna llamada `back-tier`.

```
Tu Nombre  jueves 29 enero 2026 11:28
[root@server2asir usuario]$mkdir /prometheus
Tu Nombre  jueves 29 enero 2026 11:28
[root@server2asir usuario]$cd /prometheus/
Tu Nombre  jueves 29 enero 2026 11:28
```

nano docker-compose.yml

```
GNU nano 6.2
version: '3.8'

services:
  # Capa de Interfaz
  frontend:
    image: nginx:alpine
    container_name: frontend-service
    ports:
      - "8080:80"
    networks:
      - back-tier
    depends_on:
      - backend

  # Capa de Aplicación
  backend:
    image: katacoda/docker-http-server
    container_name: backend-service
    networks:
      - back-tier
    depends_on:
      - db

  # Capa de Datos
  db:
    image: redis:alpine
    container_name: redis-db
    networks:
      - back-tier

networks:
  back-tier:
    driver: bridge
```

version: '3.8'

services:

Capa de Interfaz

frontend:

image: nginx:alpine

container_name: frontend-service

ports:

- "8080:80"

networks:

- back-tier

depends_on:

- backend

Capa de Aplicación

backend:

image: katacoda/docker-http-server

container_name: backend-service

networks:

- back-tier

depends_on:

- db

Capa de Datos

db:

image: redis:alpine

container_name: redis-db

networks:

- back-tier

networks:

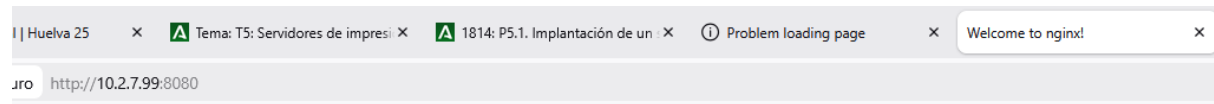
back-tier:

driver: bridge

```
[root@server2asir prometheus]# docker compose up -d
WARN[0000] /prometheus/docker-compose.yml: the attribute 'version' is obsolete, it will be ignored, please remove it to avoid potential confusion
[*] Running 19/19
✓Frontend Pulled 10.8s
✓589082ba8eae Pull complete 2.0s
✓9331cd6829ab Pull complete 2.9s
✓211bae0ea56 Pull complete 2.9s
✓9739627526d7 Pull complete 2.9s
✓6c2fac63521d Pull complete 3.0s
✓76f7e65f63b0 Pull complete 3.0s
✓cd835559902a Pull complete 3.2s
✓55cbbd6285fe Pull complete 4.9s
✓backend Pulled 0.4s
✓f199a4722ae Pull complete 0.8s
✓db Pulled 9.2s
✓d49a2dee86fb Pull complete 1.1s
✓b82811f7b773 Pull complete 1.1s
✓33b2a99c70a5 Pull complete 1.5s
✓fa398079aad5 Pull complete 3.2s
✓6c55850f77u4 Pull complete 3.2s
✓d4f4b70def5a Pull complete 3.2s
✓c137550d2aef Pull complete 3.3s
[*] Running 0/0
✓Network prometheus_back-tier Created 0.1s
✓Container redis-db Started 0.6s
✓Container backend-service Started 0.5s
✓Container frontend-service Started 1.2s
Tu Nombre jueves 29 enero 2026 11:30
[root@server2asir prometheus]#
```

Comprobamos que funciona:

<http://10.2.7.99:8080>



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

3. Fase 2: El Stack de Monitorización

Ahora que la app "negocio" funciona, vamos a montar el sistema de vigilancia.

1. **Arquitectura Prometheus:** Añade Prometheus al `docker-compose.yml`.
 - Debes crear un archivo de configuración `prometheus.yml`.
 - Configura el `scrape_interval` a 5 segundos para que las gráficas sean fluidas en clase.
2. **Despliegue de Exporters (Los "Agentes"):** Prometheus no sabe leer la CPU del host por sí solo. Debes añadir:
 - **Node Exporter:** Para métricas de la máquina virtual (Host).
 - **cAdvisor:** Para métricas específicas de cada contenedor (memoria usada por Redis, CPU del Nginx, etc.).

En la misma carpeta donde tenemos el `docker-compose.yml`, creamos el archivo `prometheus.yml`

```
GNU nano 6.2
global:
  scrape_interval: 5s
|
scrape_configs:
  - job_name: 'prometheus'
    static_configs:
      - targets: ['localhost:9090']

  - job_name: 'node-exporter'
    static_configs:
      - targets: ['node-exporter:9100']

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

Añadimos esto al docker-compose.yml

```
prometheus:
  image: prom/prometheus:latest
  container_name: prometheus
  volumes:
    - ./prometheus.yml:/etc/prometheus/prometheus.yml
  ports:
    - "9090:9090"
  networks:
    - back-tier

node-exporter:
  image: prom/node-exporter:latest
  container_name: node-exporter
  volumes:
    - /proc:/host/proc:ro
    - /sys:/host/sys:ro
    - /:/rootfs:ro
  command:
    - '--path.procfs=/host/proc'
    - '--path.sysfs=/host/sys'
    - '--collector.filesystem.ignored-mount-points=^/(sys|proc|dev|host|etc)($|/)'
  networks:
    - back-tier

cadvisor:
  image: gcr.io/cadvisor/cadvisor:latest
  container_name: cadvisor
  volumes:
    - /:/rootfs:ro
    - /var/run:/var/run:rw
    - /sys:/sys:ro
    - /var/lib/docker:/var/lib/docker:ro
  networks:
    - back-tier

networks:
  back-tier:
    driver: bridge
```

NOTA: Se ha añadido “restart: unless-stopped” a los contenedores para no tener que levantarlos manualmente cada vez que iniciemos el servidor.

3. **Verificación:** Accede a <http://localhost:9090> y comprueba en el menú *Status -> Targets* que todos los componentes aparecen en verde (**UP**).

<http://10.2.7.99:9090/targets>

The screenshot shows the Prometheus web interface at the URL <http://10.2.7.99:9090/targets>. The interface has a dark header with the Prometheus logo, navigation links (Query, Alerts, Status > Target health), and user controls (Sign in, settings, etc.). Below the header, there are filters for 'Select scrape pool', 'Filter by target health', and 'Filter by endpoint or labels'. The main content area displays three target groups, each with a table of targets.

Target Group	Endpoint	Labels	Last scrape	State
cadvisor	http://cadvisor:8080/metrics	instance="cadvisor:8080" job="cadvisor"	-729ms ago 341ms	UP
	1 / 1 up			
node-exporter	http://node-exporter:9100/metrics	instance="node-exporter:9100" job="node-exporter"	1.453s ago 54ms	UP
	1 / 1 up			
prometheus	http://localhost:9090/metrics	instance="localhost:9090" job="prometheus"	817ms ago 12ms	UP
	1 / 1 up			