09 - Healthchecks y Logging

Monitoreo y gestión de logs

Objetivo

Aprender a implementar healthchecks y gestionar logs en contenedores Docker.

¿Qué aprenderás?

- Implementar healthchecks en Dockerfiles
- Configurar healthchecks en Docker Compose
- Gestionar logs de contenedores
- Rotación de logs

¿Qué son los Healthchecks?

Permiten a Docker verificar si un contenedor está funcionando correctamente.

Analogía: Como un médico que revisa el pulso periódicamente.

Sin healthcheck: Docker asume que el contenedor está bien si está corriendo
Con healthcheck: Docker verifica que realmente funcione

¿Por qué son Importantes?

✅ **Detección temprana**: Saber si algo falla
✅ **Dependencias**: Esperar a que servicios estén listos
✅ **Auto-recuperación**: Reiniciar si falla
✅ **Monitoreo**: Estado claro (healthy/unhealthy)

Healthcheck en Dockerfile

```
FROM python:3.9-slim

WORKDIR /app
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt
COPY app.py .

EXPOSE 5000

# Healthcheck
HEALTHCHECK --interval=30s --timeout=3s --start-period=5s --retries=3 \
  CMD curl -f http://localhost:5000/health || exit 1

CMD ["python", "app.py"]
```

Parámetros del Healthcheck

Parámetro	Descripción	Default
`--interval`	Tiempo entre checks	30s
`--timeout`	Tiempo máximo de espera	30s
`--start-period`	Tiempo inicial sin fallos	0s
`--retries`	Intentos antes de unhealthy	3

Aplicación con Endpoint de Health

```
from flask import Flask, jsonify
import time

app = Flask(__name__)
start_time = time.time()

@app.route('/health')
def health():
    uptime = time.time() - start_time
    return jsonify({
        'status': 'healthy',
        'uptime': uptime
    }), 200
```

Verificar Healthcheck

```
# Ver estado del healthcheck
docker ps  # Muestra (healthy) o (unhealthy)

# Ver detalles
docker inspect --format='{{json .State.Health}}' app | python -m json.tool
```

Healthcheck en Docker Compose

```yaml
services:
  web:
    build: ./app
    healthcheck:
      test: ["CMD", "curl", "-f", "http://localhost:5000/health"]
      interval: 30s
      timeout: 10s
      retries: 3
      start_period: 40s

  db:
    image: postgres:15
    healthcheck:
      test: ["CMD-SHELL", "pg_isready -U postgres"]
      interval: 10s
      timeout: 5s
      retries: 5
```

Dependencias con Healthchecks

```yaml
services:
  web:
    depends_on:
      db:
        condition: service_healthy  # Espera a que db esté healthy
```

Ver Logs Básicos

```
# Ver logs de un contenedor
docker logs <container_id>

# Seguir logs en tiempo real
docker logs -f <container_id>

# Últimas N líneas
docker logs --tail 100 <container_id>

# Logs con timestamp
docker logs -t <container_id>
```

Logging en Docker Compose

```yaml
services:
  web:
    build: ./app
    logging:
      driver: "json-file"
      options:
        max-size: "10m"
        max-file: "3"
        labels: "production"
```

Logging Estructurado

```
import json
import logging
import sys

class JSONFormatter(logging.Formatter):
    def format(self, record):
        log_entry = {
            'timestamp': self.formatTime(record),
            'level': record.levelname,
            'message': record.getMessage(),
        }
        return json.dumps(log_entry)

logger = logging.getLogger()
handler = logging.StreamHandler(sys.stdout)
handler.setFormatter(JSONFormatter())
logger.addHandler(handler)
```

Comandos Útiles

```
# Ver logs de Docker Compose
docker compose logs
docker compose logs -f web

# Ver eventos de contenedor
docker events

# Inspeccionar configuración de logging
docker inspect --format='{{json .HostConfig.LogConfig}}' <container>
```

Mejores Prácticas

✅ Implementa healthchecks en todas las aplicaciones
✅ Usa endpoints `/health` o `/healthz`
✅ Configura rotación de logs (max-size, max-file)
✅ Usa logging estructurado (JSON)
✅ Centraliza logs en producción
✅ Monitorea el estado de healthchecks

Práctica

1. Agrega un healthcheck a tu aplicación
2. Configura logging con rotación
3. Verifica el estado de healthchecks
4. Implementa logging estructurado

Siguiente Paso

Aprende mejores prácticas para producción y despliegue.

Módulo 10: Producción y Mejores Prácticas

Preguntas?

¡Tiempo para preguntas y práctica!