Guía del Estudiante - Video 2

Despliegue de Moodle con Docker Compose + Observabilidad IA

Código utilizado en el video

1. Objetivo

Desplegar una infraestructura completa con Moodle, base de datos MariaDB, monitoreo con Netdata, servidor de modelos LLM (Ollama) y análisis inteligente de logs, todo orquestado con Docker Compose.

2. Estructura del proyecto

```
proyecto/
          docker-compose.yml
          .env
          loginsights-docker/
                Dockerfile
                loginsights.py
                entrypoint.sh
```

3. Archivo docker-compose.yml

```
version: "3.9"
services:
  #1) Base de datos MariaDB
  db:
    image: mariadb:10.11
    container_name: moodle-db
    restart: unless-stopped
    env_file: .env
    healthcheck:
      test: ["CMD-SHELL", "mysqladmin ping -h 127.0.0.1 -uroot -p$${
    MARIADB_ROOT_PASSWORD} --silent"]
      interval: 10s
      retries: 5
      start_period: 30s
    environment:
      MARIADB_DATABASE: ${MOODLE_DB_NAME}
      MARIADB_USER:     ${MOODLE_DB_USER}
      MARIADB_PASSWORD: ${MOODLE_DB_PASSWORD}
      MARIADB_ROOT_PASSWORD: ${MARIADB_ROOT_PASSWORD}
    command: >
      --transaction-isolation=READ-COMMITTED
      --innodb_buffer_pool_size=256M
      --character-set-server=utf8mb4
      --collation-server=utf8mb4_unicode_ci
    volumes:
      - db_data:/var/lib/mysql
    networks:
      - moodle_network
    logging:
      driver: json-file
      options:
        max-size: "50m"
```

```
      max-file: "2"

# 2) Moodle (Bitnami)
moodle:
  image: bitnami/moodle:latest
  container_name: moodle-app
  restart: unless-stopped
  depends_on:
    db:
      condition: service_healthy
  env_file: .env
  environment:
    ALLOW_EMPTY_PASSWORD: "no"
    MOODLE_DATABASE_TYPE: mariadb
    MOODLE_DATABASE_HOST: db
    MOODLE_DATABASE_PORT_NUMBER: 3306
    MOODLE_DATABASE_USER: ${MOODLE_DB_USER}
    MOODLE_DATABASE_PASSWORD: ${MOODLE_DB_PASSWORD}
    MOODLE_DATABASE_NAME: ${MOODLE_DB_NAME}
    MOODLE_USERNAME: ${MOODLE_USERNAME}
    MOODLE_PASSWORD: ${MOODLE_PASSWORD}
    MOODLE_SITE_NAME: ${MOODLE_SITE_NAME}
    BITNAMI_DEBUG: "true"
    APACHE_HTTPD_LOG_LEVEL: debug
    MOODLE_DEBUG: "38911"
    MOODLE_DEBUG_DISPLAY: "true"
    PHP_ERROR_REPORTING: "E_ALL"
  ports:
    - "8080:8080"
    - "8443:8443"
  volumes:
    - moodle_data:/bitnami/moodledata
    - moodle_install:/bitnami/moodle
    - moodle_apache_conf:/bitnami/apache
  networks:
    - moodle_network
  logging:
    driver: json-file
    options:
      max-size: "100m"
      max-file: "3"

# 3) Netdata con IA
netdata:
  image: netdata/netdata:stable
  container_name: netdata
  network_mode: host
  pid: host
  cap_add: [SYS_PTRACE]
  security_opt: [apparmor:unconfined]
  environment:
    - NETDATA_ML=yes
  volumes:
    - netdata_lib:/var/lib/netdata
    - netdata_cache:/var/cache/netdata
    - /etc/passwd:/host/etc/passwd:ro
    - /proc:/host/proc:ro
    - /sys:/host/sys:ro
  restart: unless-stopped

# 4) Ollama (servidor LLM local)
ollama:
  image: ollama/ollama:latest
```

```
    container_name: ollama
    command: ["serve"]
    ports:
      - "11434:11434"
    volumes:
      - ollama_models:/root/.ollama
    restart: unless-stopped
    networks:
      - moodle_network
    logging:
      driver: json-file
      options:
        max-size: "50m"
        max-file: "2"
    environment:
      - OLLAMA_HOST=0.0.0.0:11434
      - OLLAMA_NUM_PARALLEL=1
      - OLLAMA_MAX_LOADED_MODELS=1
      - OLLAMA_KEEP_ALIVE=5m

  # 5) LogInsights (an lisis inteligente de logs con LLM)
  loginsights:
    build: ./loginsights-docker
    container_name: loginsights
    depends_on:
      - ollama
      - moodle
      - db
    volumes:
      - log_reports:/reports
      - /var/run/docker.sock:/var/run/docker.sock:ro
    environment:
      - OLLAMA_HOST=http://ollama:11434
      - LOG_LEVEL=DEBUG
      - CONTAINER_NAMES=moodle-app,ollama,moodle-db
      - MODEL=tinyllama:1.1b
      - INTERVAL=120
      - ANALYSIS_TIMEOUT=180
    networks:
      - moodle_network
    restart: unless-stopped
    logging:
      driver: json-file
      options:
        max-size: "50m"
        max-file: "2"

networks:
  moodle_network:
    driver: bridge

volumes:
  db_data:
  moodle_data:
  moodle_install:
  moodle_apache_conf:
  netdata_lib:
  netdata_cache:
  ollama_models:
  log_reports:
```

4. Dockerfile para LogInsights

```
FROM python:3.11-slim

# 1) Dependencias b sicas + docker CLI
RUN apt-get update \
    && apt-get install -y --no-install-recommends \
    curl \
    jq \
    docker.io \
    && rm -rf /var/lib/apt/lists/*

# 2) Instalar dependencias de Python
RUN pip install --no-cache-dir \
    docker \
    requests \
    pyyaml \
    rich

# 3) Copiar scripts
COPY entrypoint.sh /entrypoint.sh
COPY loginsights.py /loginsights.py
RUN chmod +x /entrypoint.sh

# 4) Directorio de reportes
RUN mkdir -p /reports

ENV PYTHONUNBUFFERED=1
ENV OLLAMA_HOST=http://ollama:11434
ENV LOG_LEVEL=INFO

WORKDIR /
ENTRYPOINT ["/entrypoint.sh"]
```

5. Script Python loginsights.py

```
#!/usr/bin/env python3
"""
LogInsights - Sistema de an lisis inteligente de logs con LLM
- Recolecta logs de contenedores Docker
- Analiza con modelos de lenguaje usando Ollama
- Genera reportes estructurados en /reports
"""
import os
import time
from datetime import datetime
from pathlib import Path
import docker
import requests

#
    Configuraci n


OLLAMA_HOST = os.getenv("OLLAMA_HOST", "http://ollama:11434")
MODEL       = os.getenv("MODEL", "tinyllama:1.1b")
INTERVAL    = int(os.getenv("INTERVAL", "120"))
ANAL_TIMEOUT = int(os.getenv("ANALYSIS_TIMEOUT", "180"))
CONTAINERS   = [c.strip() for c in os.getenv("CONTAINER_NAMES", "moodle-app").
    split(",")]
```

```
LOG_LEVEL    = os.getenv("LOG_LEVEL", "INFO")

#    Cliente Docker

try:
    docker_client = docker.DockerClient(base_url="unix:///var/run/docker.sock")
    docker_client.ping()
except Exception as exc:
    print(f"    Error conectando a Docker: {exc}")
    exit(1)

#    Funciones auxiliares

def get_container_status(name: str) -> str:
    try:
        return docker_client.containers.get(name).status
    except docker.errors.NotFound:
        return "not_found"
    except Exception as exc:
        print(f"        Estado de {name}: {exc}")
        return "error"

def get_recent_logs(name: str, lines: int = 100) -> str:
    try:
        cont = docker_client.containers.get(name)
        return cont.logs(tail=lines, timestamps=True).decode("utf-8")
    except Exception as exc:
        return f"Error obteniendo logs: {exc}"

def analyze_with_ollama(text: str, container: str) -> str:
    """
    Llama a /api/generate de Ollama para an lisis inteligente de logs
    """
    prompt = f"""Analiza los siguientes logs del contenedor **{container}** y
    genera un resumen:
1. Mensajes m s relevantes
2. Errores o advertencias cr ticas
3. Estado general del servicio
4. Acciones recomendadas
Responde en espa ol de forma breve y estructurada.
Logs:
{text[:4000]}"""  # se limita para no saturar al modelo

    try:
        resp = requests.post(
            f"{OLLAMA_HOST}/api/generate",
            json={
                "model": MODEL,
                "prompt": prompt,
                "stream": False,
                "options": {
                    "temperature": 0.4,
                    "num_predict": 512
                },
            },
            timeout=ANAL_TIMEOUT,
        )
        if resp.status_code == 200:
            return resp.json().get("response", "Respuesta vac a")
        return f"Error {resp.status_code}: {resp.text}"
    except requests.exceptions.Timeout:
```

```
        return "       Timeout alcanzado durante la llamada a Ollama"
    except Exception as exc:
        return f"    Error llamando a Ollama: {exc}"

def save_report(container: str, analysis: str, logs: str) -> None:
    ts = datetime.now().strftime("%Y%m%d_%H%M%S")
    path = Path(f"/reports/summary_{container}_{ts}.txt")
    with path.open("w") as f:
        f.write(f"=== LogInsights - An lisis de logs para {container} ===\n")
        f.write(f"Timestamp: {datetime.now().isoformat()}\n")
        f.write(f"Estado del contenedor: {get_container_status(container)}\n")
        f.write(f"Modelo usado: {MODEL}\n")
        f.write("=" * 50 + "\n\n")
        f.write("=== AN LISIS ===\n")
        f.write(analysis + "\n\n")
        f.write("=== LOGS ORIGINALES ( ltimas  50 l neas) ===\n")
        for line in logs.splitlines()[-50:]:
            f.write(line + "\n")
    print(f"    Reporte guardado: {path}")

def list_last_reports() -> None:
    rep_dir = Path("/reports")
    if not rep_dir.exists():
        return
    reports = sorted(rep_dir.glob("summary_*.txt"))[-10:]
    if reports:
        print("\  n      ltimos  reportes:")
        for rep in reports:
            print(f"      {rep.name} ({rep.stat().st_size/1024:.1f} KB)")

#

     Main loop


if __name__ == "__main__":
    print("       LogInsights - Sistema de an lisis inteligente de logs")
    print(f"   Contenedores: {', '.join(CONTAINERS)}")
    print(f"   Modelo: {MODEL} / Timeout por request: {ANAL_TIMEOUT}s\n")

    Path("/reports").mkdir(exist_ok=True)

    # Peque a espera inicial
    time.sleep(10)

    while True:
        print(f"\  n     {datetime.now():%Y-%m-%d %H:%M:%S}     nuevo ciclo")
        for cont in CONTAINERS:
            if get_container_status(cont) == "running":
                logs = get_recent_logs(cont, 100)
                result = analyze_with_ollama(logs, cont)
                save_report(cont, result, logs)
            else:
                print(f"        {cont} no est  en estado running")

        list_last_reports()
        print(f"\  n     Esperando {INTERVAL} s  ")
        time.sleep(INTERVAL)
```

6. Comandos de ejecución

```
# 1. Crear el archivo .env con las variables necesarias
$ nano .env

# 2. Construir y levantar todos los servicios
$ docker-compose up -d

# 3. Verificar el estado de los contenedores
$ docker-compose ps

# 4. Ver logs en tiempo real
$ docker-compose logs -f

# 5. Acceder a Moodle
$ curl http://localhost:8080

# 6. Acceder a Netdata
$ curl http://localhost:19999

# 7. Verificar Ollama
$ curl http://localhost:11434/api/tags

# 8. Ver reportes de LogInsights
$ docker exec loginsights ls -la /reports/
```