App de Traducción Orquestada con Docker Compose y Swarm

Estudiantes:

• Leidis Lopez

• Camilo Perez

• Santiago Muñoz

Repositorio de GitHub: https://github.com/ceperezm/App-traductor-orquestada-docker-compose-swarm.git

Dockerhub: https://hub.docker.com/repository/docker/ceperezm/traductor-genai

Arquitectura General

La arquitectura del proyecto está compuesta por dos servicios principales: el servidor MLflow, expuesto en el puerto 5000, y la aplicación de traducción desarrollada con Gradio, accesible desde el puerto 7860, la aplicación Gradio integra la lógica de traducción. En Docker Compose ambos servicios se ejecutan en una red bridge local, mientras que en Docker Swarm funcionan sobre una red overlay distribuida. Los volúmenes mlflow-db-data y mlflow-artifacts-data almacenan de forma persistente la base de datos y los artifacts del servidor MLflow.

Diferencias entre docker-compose.yml y docker-stack.yml

En docker-compose.yml la aplicación se construye localmente mediante build: context: ., utiliza una sola réplica del servicio app-traductor, emplea la red traductor-net con driver bridge y monta los volúmenes persistentes de MLflow.
En docker-stack.yml se despliega a partir de la imagen ya construida ceperezm/traductor-genai:1.0.0, se inician dos réplicas del servicio, se usa la red traductor-net con driver overlay y attachable: true, y se define una configuración de despliegue completa que incluye restart_policy, update_config y rollback_config.

Comandos Principales Usados

Los comandos utilizados para el flujo de trabajo fueron:

- docker-compose up --build para levantar los servicios localmente.
- docker-compose down para detenerlos.

- docker push usuario/imagen:tag para publicar imágenes.
- docker swarm init para iniciar el clúster.
- docker stack deploy -c docker-stack.yml app para desplegar el stack completo.
- docker service ls para listar los servicios en Swarm.
- docker service scale app_traductor=X para ajustar el número de réplicas.

Observaciones sobre Latencia y Calidad de Traducción

En general, Docker Compose ofrece menor latencia debido a que todos los servicios operan dentro de la misma máquina. En Docker Swarm, la latencia aumenta ligeramente por el enrutamiento interno entre nodos, aunque se mantiene estable incluso al escalar réplicas del servicio. La calidad de traducción se mantuvo constante en todos los casos, utilizando modelos como Google Gemma, Gemini Flash, Qwen y DeepSeek en sus versiones gratuitas. Todas las traducciones fueron registradas en MLflow junto con métricas de latencia, longitud del texto y artifacts JSON. El sistema se comportó de manera estable, con resultados consistentes tanto en Compose como en Swarm.