Guía Integral 2025 para el Despliegue de Modelos de ML, DL y LLM

Incluye Tarea para Estudiantes

Aldo Camargo

2025

Resumen

Esta guía integra las mejores prácticas modernas para el despliegue de modelos de *Machine Learning* (ML), *Deep Learning* (DL) y *Large Language Models* (LLM). Incluye contenedorización, orquestación, optimización, seguridad y operación en producción. Al final del documento se incluye la tarea asignada a los estudiantes.

Índice

1. Introducción

El despliegue de modelos de ML, DL y LLM en aplicaciones reales implica el uso de diversas herramientas y el cumplimiento de criterios esenciales de seguridad, eficiencia y escalabilidad. Tecnologías como Docker y Kubernetes constituyen la base de estos sistemas modernos de producción.

A lo largo de este documento se integran prácticas operativas actuales, enfoques de seguridad, técnicas de optimización y patrones de arquitectura necesarios para implementar sistemas de IA robustos.

2. Contenedorización con Docker

Docker es el estándar para empaquetar modelos con sus dependencias, asegurando portabilidad y ejecución consistente entre entornos de desarrollo, pruebas y producción.

2.1. Buenas prácticas para imágenes

Imágenes base mínimas

Utilizar imágenes minimizadas como:

- `python:3.10-slim`
- `distroless`

Esto reduce el tamaño total de la imagen y su superficie de ataque.

Compilación multinivel (Multi-stage Builds)

```
# Build Stage
FROM python:3.10-builder as builder
RUN pip install --user requirements.txt

# Runtime Stage
FROM python:3.10-slim
COPY --from=builder /root/.local /root/.local
ENV PATH=/root/.local/bin:$PATH
COPY . /app
CMD ["uvicorn", "main:app", "--host", "0.0.0.0"]
```

Soporte GPU

Use imágenes CUDA únicamente cuando sea necesario, por ejemplo:

- `nvidia/cuda:12.x.x-base-ubuntu22.04`

Para inferencia en CPU utilice imágenes Linux estándar para reducir peso y complejidad.

2.2. Manejo de pesos del modelo

- **Modelos pequeños (<500MB):** Se pueden incluir en la imagen Docker.
- **Modelos grandes o LLMs:** No deben incluirse en la imagen. Deben montarse como volúmenes o descargarse al arrancar mediante un *init container*.

3. Orquestación con Kubernetes

Kubernetes (K8s) es la plataforma estándar para desplegar cargas de IA distribuidas a gran escala.

3.1. Componentes clave para ML/DL

- **Deployments:** para inferencia sin estado.
- **StatefulSets:** para sesiones pegajosas o cachés persistentes.
- **HPA (Horizontal Pod Autoscaler):** escala por CPU/memoria.
- **KEDA:** escala por métricas personalizadas como *queue depth* o uso de GPU.

3.2. Asignación de cargas GPU

- `nodeSelector` para seleccionar nodos GPU.
- **Taints/Tolerations** para evitar que cargas no-GPU ocupen nodos costosos.
- `affinity` para agrupar cargas relacionadas.

3.3. Frameworks de *serving* sobre Kubernetes

KServe

- Autoscaling (incluye *scale-to-zero*).
- Rollouts canarios.
- Soporte para protocolos de inferencia estandarizados.

Ray Serve

Ideal para pipelines complejos como RAG:

- modelo de embeddings,
- búsqueda vectorial,
- generación con LLM.

4. Despliegue de Modelos de Machine Learning

El despliegue de modelos de ML consiste en convertir prototipos en sistemas de producción. Incluye soporte para APIs, escalabilidad, actualización continua y monitoreo.

4.1. Estrategias comunes

- Contenedorización con Docker y Kubernetes.
- Exposición mediante servidores de modelos.
- *Serverless* para tráfico esporádico.
- Procesamiento batch.
- Inferencia en dispositivos edge usando ONNX o TensorFlow Lite.

5. Despliegue de Modelos de Deep Learning

Los modelos de Deep Learning requieren técnicas adicionales por su alto costo computacional.

5.1. Optimización

- Compresión de modelos.
- Cuantización (4-bit, 8-bit).
- *Knowledge distillation.*

5.2. Frameworks de inferencia

- TensorFlow Serving.
- TorchServe.
- NVIDIA Triton Inference Server.

6. Despliegue de Modelos LLM

Los modelos de lenguaje extenso requieren infraestructuras especializadas.

6.1. Motores de Servido

- **vLLM** (alto rendimiento).
- **TGI** (robusto y flexible).
- **TensorRT-LLM** (máximo desempeño en GPU NVIDIA).

6.2. Técnicas de Optimización

- Cuantización: AWQ, GPTQ.
- *Continuous batching.*
- Adaptadores LoRA dinámicos.

7. Criterios de Seguridad

7.1. Seguridad en Contenedores

- Escaneo de vulnerabilidades: Trivy, Grype.
- Usuario no-root.
- Firmado de imágenes: Cosign/Sigstore.

7.2. Seguridad en Infraestructura

- Políticas de red estrictas.
- Límites de recursos para prevenir DoS.
- Manejo seguro de secretos.

7.3. Seguridad para LLM

- Defensa contra Prompt Injection.
- Redacción de datos sensibles.
- Uso de formatos seguros (`safetensors`).

8. Monitoreo y Respuesta a Incidentes

- Observabilidad del modelo.
- Alertas basadas en latencia, precisión y fallos.
- Auditorías de seguridad periódicas.

9. Lista de Verificación

Docker

- Multi-stage builds.
- Usuario no-root.
- Imagen base mínima.

Kubernetes

- KServe o Ray Serve.
- KEDA para autoscaling avanzado.

LLM

- vLLM o TGI.
- Cuantización.
- Formato seguro: `safetensors`.

Seguridad

- Escaneo de vulnerabilidades.
- Network Policies.
- Guardrails contra prompt injection.

Tarea

Tarea:
Desarrollar un plan de despliegue del modelo o los modelos que utilizarán en su Tesis de Maestría. Incluyan la mayor cantidad posible de los puntos revisados en la clase de hoy.

Fecha de entrega: 10 de diciembre de 2025.