Caso práctico 4: Monitoreo de caso real con TensorFlow Extended.

Luis Fabian Amagua

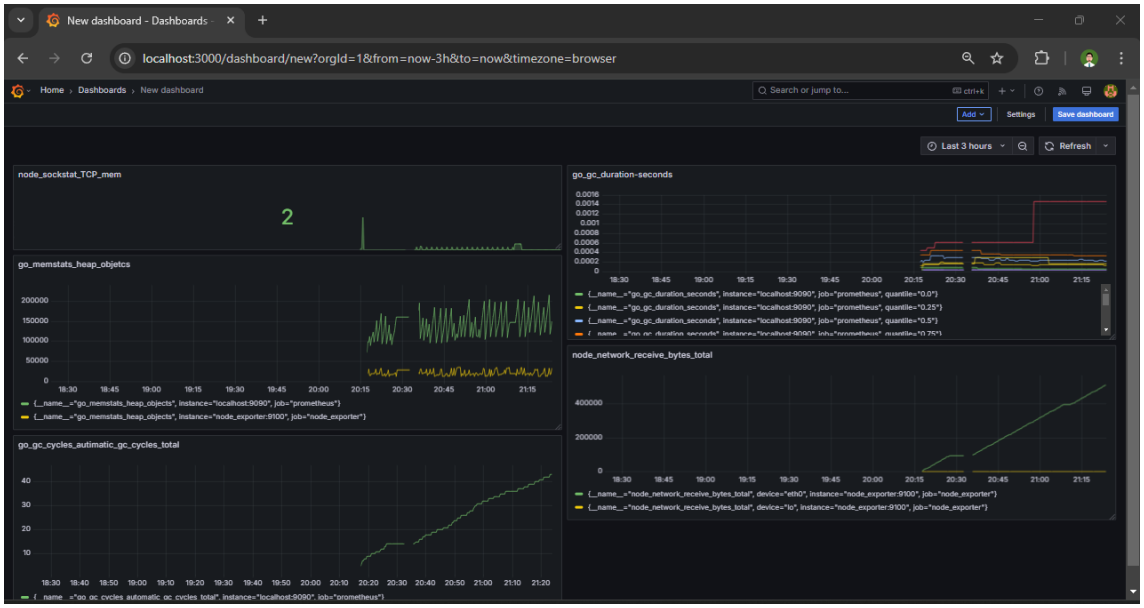
Realizar el monitoreo utilizando TensorFlow Extended (TFX) con un caso real. El objetivo principal de este taller es familiarizar a los estudiantes con las técnicas y herramientas de monitoreo aplicadas a pipelines de aprendizaje automático utilizando TFX y Grafana.

El proceso de integración comenzó con el desarrollo de un modelo de predicción de precios de casas utilizando Python. Este modelo fue diseñado para prever valores futuros de propiedades inmobiliarias en función de datos históricos y características relevantes de las viviendas. Una vez que el modelo fue entrenado y probado, se decidió implementar la solución en un entorno de contenedores usando Docker para facilitar su despliegue y ejecución de manera eficiente y escalable.

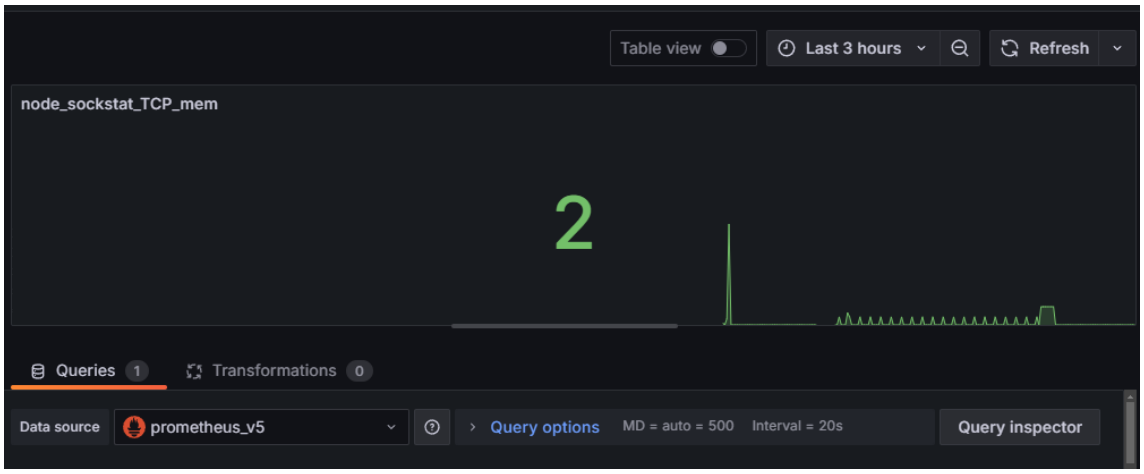
Se siguieron los siguientes pasos:

1. Preparación del Entorno Docker
   Para iniciar el proceso de contenedorización, se crearon los archivos necesarios (Dockerfile y otros scripts relacionados) que definen las dependencias, bibliotecas y configuraciones necesarias para ejecutar el modelo de predicción en un contenedor. Docker se utilizó para asegurar que el entorno fuera replicable, portátil y libre de conflictos, lo cual facilita tanto la implementación como la escalabilidad.
2. Cargar el Modelo a Docker
   Los archivos Python (.py) que contienen el modelo de predicción fueron integrados dentro de la imagen de Docker. Esta imagen fue luego construida y desplegada en el contenedor. El contenedor, a su vez, ejecutó el modelo en el ambiente aislado, garantizando que las dependencias y el entorno fueran consistentes durante la ejecución.
3. Exposición de Métricas
   Una vez que el modelo estaba funcionando dentro del contenedor, se configuró un sistema para exponer métricas de rendimiento y resultados generados por el modelo, como los precios de las predicciones, el error de las predicciones, y otros indicadores relevantes. Estas métricas fueron expuestas a través de un endpoint o API que puede ser consumido por herramientas de monitoreo, como Grafana.
4. Integración con Grafana
   Con las métricas expuestas por el modelo, se integró Grafana como plataforma de visualización de datos. Grafana se conectó al endpoint de métricas, recopilando y mostrando visualizaciones en tiempo real. Las métricas relacionadas con el modelo de predicción de precios de casas, tales como los precios predichos, la precisión, y las variaciones de precio, fueron representadas mediante gráficos interactivos y paneles visuales.
5. Monitoreo Continuo y Seguimiento de Rendimiento
   A través de Grafana, se configuraron paneles de monitoreo que permiten un seguimiento en tiempo real del rendimiento del modelo. Esto incluye análisis sobre la precisión de las predicciones, la variación de precios en el tiempo y otros indicadores clave que son fundamentales para evaluar la eficacia del modelo. Estas métricas proporcionan información valiosa para ajustar y mejorar el modelo de manera continua.
6. Optimización y Escalabilidad
   El uso de Docker facilita la escalabilidad del modelo, permitiendo que múltiples instancias del contenedor se ejecuten simultáneamente, mejorando la capacidad de procesamiento y respuesta. Grafana, por su parte, puede manejar grandes volúmenes de datos y proporcionar métricas en tiempo real, lo que permite una gestión más eficaz de los recursos y un ajuste rápido en función de las métricas obtenidas.

Los resultados en grafana son los siguientes.

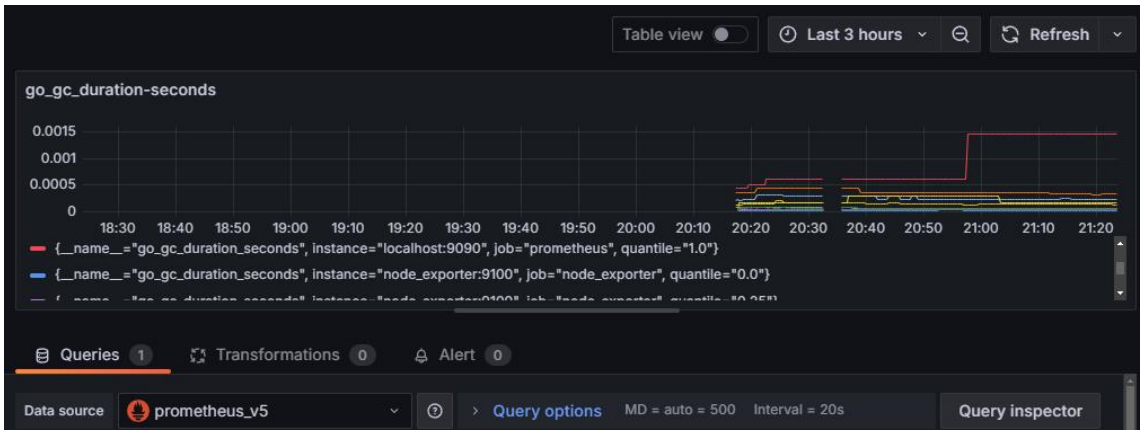


A continuación se describirá las métricas que se han usado:

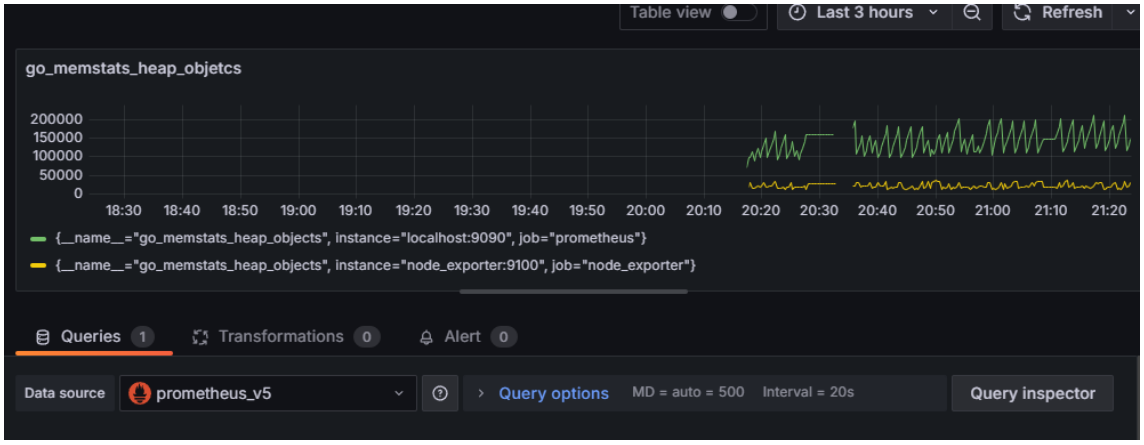


La métrica **node_sockstat_TCP_mem** indica la cantidad de memoria utilizada por las conexiones TCP en el sistema. Un valor de 2 sugiere que el uso de memoria es bajo y constante, lo cual es normal si no hay muchas conexiones activas. El pico inicial puede ser normal cuando el sistema establece nuevas conexiones TCP, lo que provoca un aumento temporal en el uso de memoria. Si el valor se mantiene en 2, indica que no hay un consumo significativo de memoria por las conexiones TCP.

Un valor constante de 2 es normal, y un aumento significativo podría señalar muchas conexiones activas, lo que requeriría investigar el tráfico de red o la gestión de conexiones.



La métrica **go_gc_duration_seconds** en Grafana mide el tiempo que tarda el garbage collector (GC) de una aplicación Go en liberar memoria. Un valor alto puede indicar problemas de rendimiento, ya que el GC está tomando más tiempo de lo esperado. Esta métrica tiene diferentes cuantiles, como quantile 1.0, que muestra el peor caso o el ciclo de GC más largo. Es normal que la gráfica de quantile 1.0 esté separada y sea más alta, ya que refleja el tiempo máximo en lugar de los promedios o valores bajos. Monitorear esta métrica te ayuda a detectar y optimizar el manejo de memoria para mejorar la eficiencia de la aplicación.



La métrica **go_memstats_heap_objects** en Grafana mide la cantidad de objetos en el heap de memoria gestionado por una aplicación Go. Monitorear esta métrica ayuda a detectar el uso de memoria y si la aplicación genera muchos objetos, lo que podría afectar el rendimiento. Un valor alto puede indicar fugas de memoria o una gestión ineficiente.

Se tienen dos lineas:

Verde (Prometheus): Muestra la memoria usada específicamente por la aplicación Go.

Amarilla (Node Exporter): Muestra el uso general de la memoria del sistema, incluyendo otros procesos.

La métrica **node_network_receive_bytes_total** en Grafana mide la cantidad total de bytes recibidos a través de las interfaces de red de un sistema. Monitorearla permite analizar el tráfico entrante y detectar picos inusuales, lo que podría indicar problemas de red o un aumento en la carga del sistema. Un valor alto o un aumento repentino puede señalar mayor actividad de red o posibles problemas.

Se tienen dos líneas:

device eth0: Mide los bytes recibidos a través de la interfaz de red principal del servidor.

device io: Mide los bytes recibidos por una interfaz secundaria o un dispositivo virtual, con un nombre más genérico.

Es normal ver estas dos métricas, ya que reflejan el tráfico de diferentes interfaces de red.

La métrica **go_gc_cycles_automatic_gc_cycles_total** en Grafana mide el número total de ciclos automáticos de recolección de basura (GC) realizados por la aplicación Go. Su crecimiento continuo es normal, ya que el GC se ejecuta constantemente para liberar memoria no utilizada.

Un valor alto no es preocupante, pero un aumento rápido puede indicar que la aplicación está generando muchos objetos o no está gestionando bien la memoria, lo que podría afectar el rendimiento.

El crecimiento continuo del contador es normal, pero un aumento excesivo podría señalar problemas de gestión de memoria que afectan el rendimiento de la aplicación.