

marvik.

Bienvenidos al curso Modelos de Lenguaje

Temario







- 23/04 - Python, FastAPI, Streamlit, Docker, Despliegue de modelos
- 30/04 - LLM, Retrieval Augmented Generation (RAG)
- 07/05 - Finetune LLM, Prompting, Buenas prácticas
- 14/05 - Entrega trabajo práctico

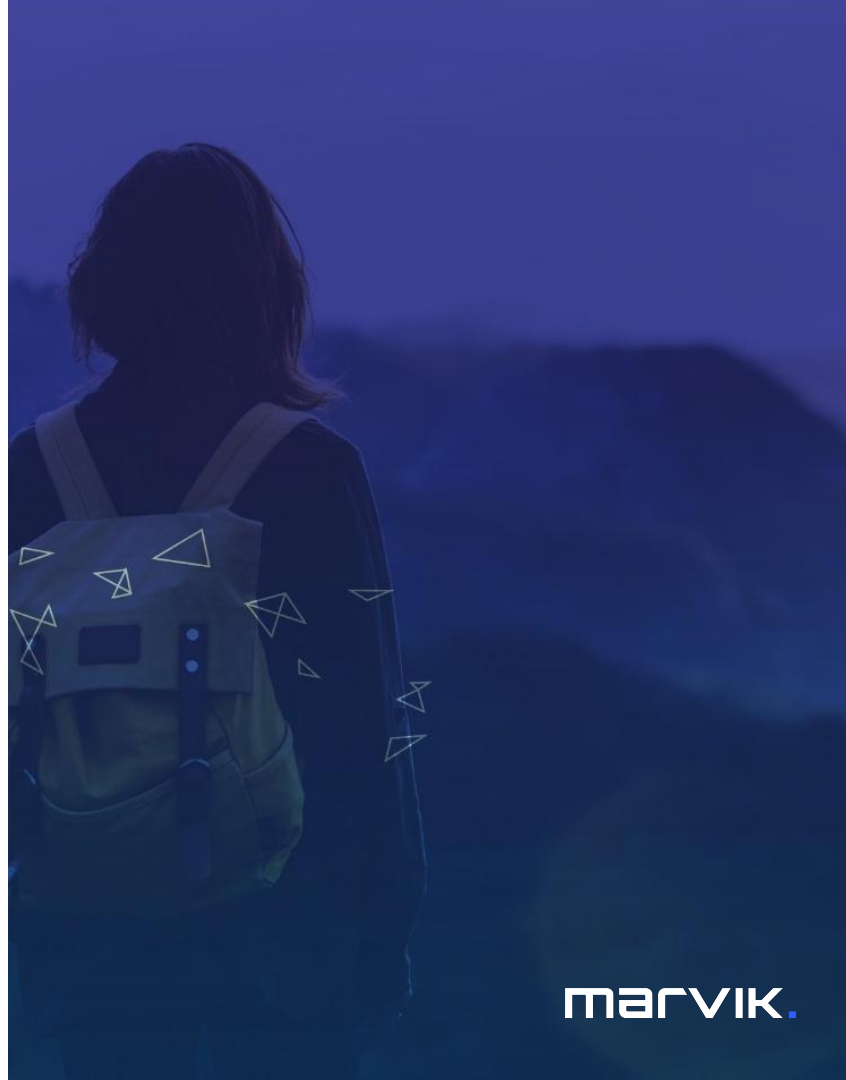


Despliegue de modelo de IA con FastAPI, Docker y Streamlit

Clase - 23/04

Agenda

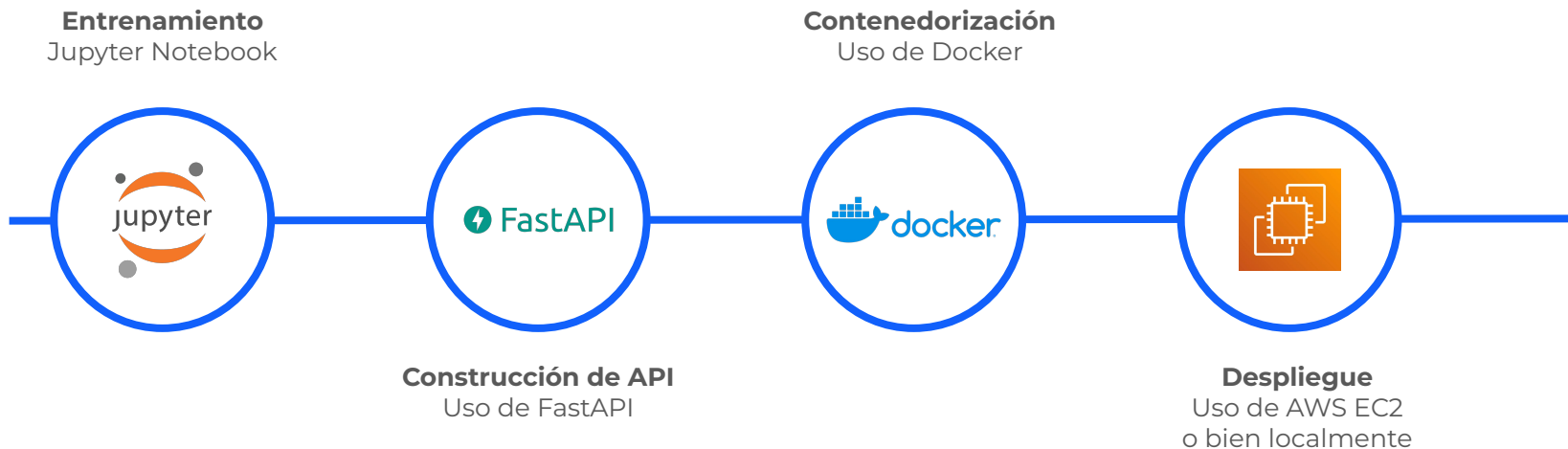
-  Introducción de conceptos y herramientas
-  Python
-  Clonación de repositorio y set up de instancia
-  Pruebas a través de API y FE
-  Despliegue con Docker
-  Buenas prácticas



Introducción de conceptos y herramientas

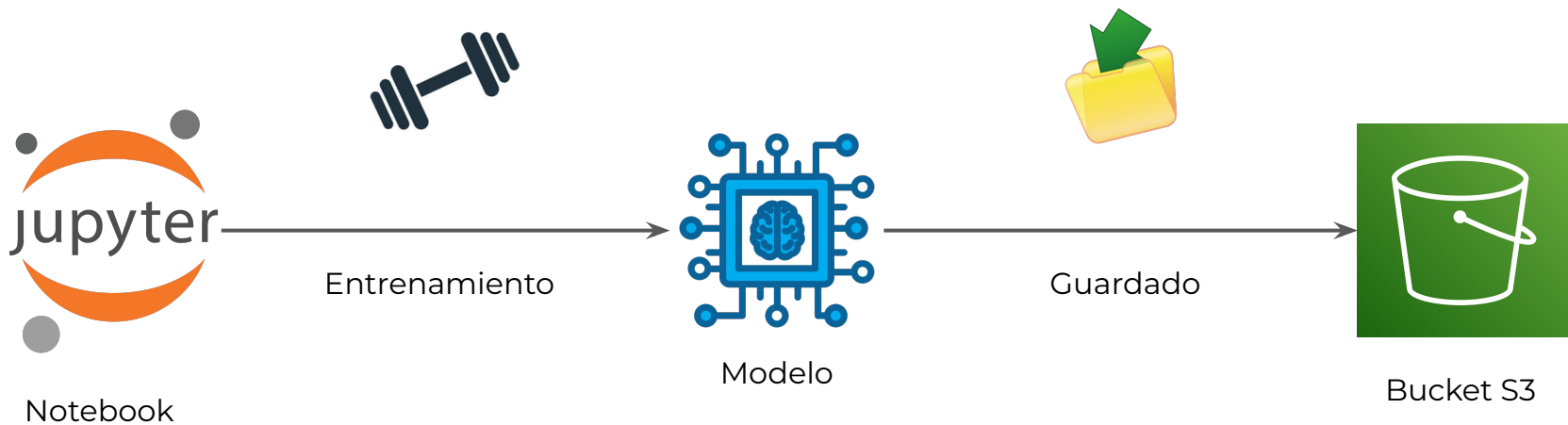
Esquema: De la notebook a la nube

marvik.



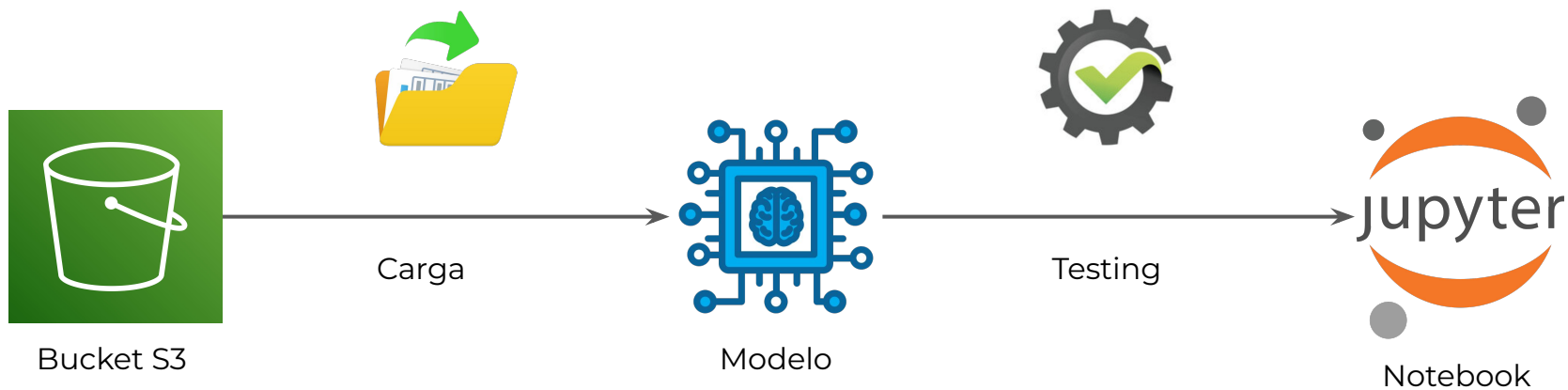
Guardado del modelo post entrenamiento

marvik.



Prueba de carga del modelo desde archivo

marvik.



La migración de archivos de Jupyter Notebook (.ipynb) a scripts de Python (.py) es un proceso esencial para la puesta en producción.

Mientras que los notebooks son excelentes para la exploración y el desarrollo interactivo, los scripts .py son más adecuados para la ejecución automatizada, la integración continua y el despliegue en entornos de producción.



¿Qué es FastAPI y uvicorn?

marvik.

FastAPI es un framework moderno y de alto rendimiento para **construir APIs con Python**.

Está diseñado para ser fácil de usar y proporcionar funcionalidades como la **validación automática de datos** y la generación de **documentación interactiva**.

FastAPI se basa en estándares abiertos como OpenAPI y **JSON Schema**, lo que facilita la interoperabilidad y la colaboración entre diferentes servicios.

Uvicorn es un servidor ASGI (Asynchronous Server Gateway Interface) rápido y ligero que se utiliza para **ejecutar aplicaciones construidas con FastAPI** permitiendo que manejen una gran cantidad de solicitudes concurrentes de manera efectiva.



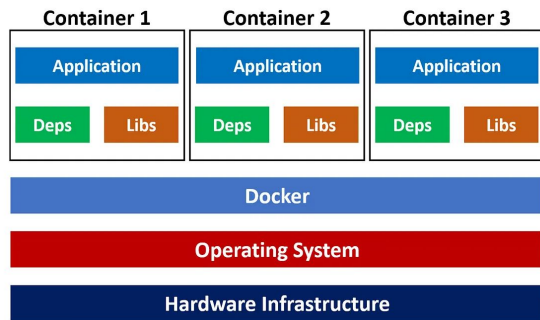
¿Qué es Docker?

marvik.

Docker es una plataforma de software que permite a los desarrolladores **construir, probar y desplegar aplicaciones** rápidamente mediante el **uso de contenedores**.

Los contenedores son **unidades portátiles** que incluyen todo lo necesario para ejecutar una aplicación: **código, runtime, bibliotecas del sistema y configuraciones**. Esto garantiza que la aplicación se **ejecute de manera consistente**, sin importar dónde se despliegue.

Docker facilita la creación de entornos de desarrollo consistentes y reproducibles, lo que **reduce los problemas de compatibilidad** entre diferentes entornos.



¿Qué es Streamlit?

marvik.

Streamlit es una herramienta de código abierto que permite a los desarrolladores **crear aplicaciones web interactivas y visualizaciones** de datos de manera rápida y sencilla, utilizando solo **código Python**.

Diseñado específicamente para científicos de datos y analistas, Streamlit facilita la **creación de interfaces de usuario** sin necesidad de conocimientos avanzados en desarrollo web.

Con Streamlit se pueden convertir sus scripts de datos en aplicaciones web interactivas con pocos comandos, lo que permite una exploración, presentación de datos e interacción con los modelos de una forma intuitiva y accesible.

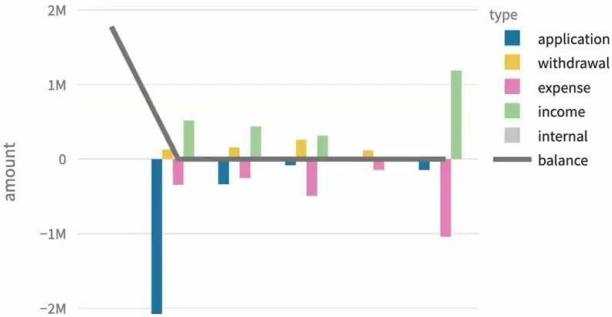


Deploy ⋮

Comparisson between Model and Baseline

	capital_loss_inst	capital_loss_proj	efficiency_inst	efficiency_proj	limit_compliance
total	R\$-4,173.32	R\$-16,129.67	57.92%	66.05%	98.15%

Transactions per day



Transactions per day



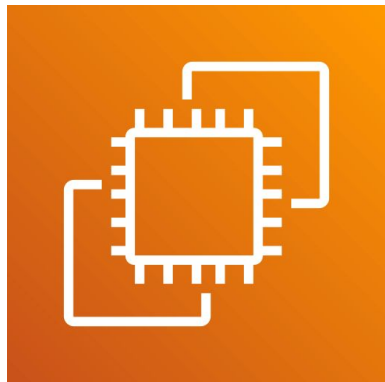
¿Qué es EC2 (Amazon Elastic Compute Cloud)?

marvik.

Amazon Elastic Compute Cloud (EC2) es un servicio web que proporciona **capacidad de cómputo** en la nube escalable permitiendo ejecutar aplicaciones y cargas de trabajo **sin necesidad de invertir en hardware**.

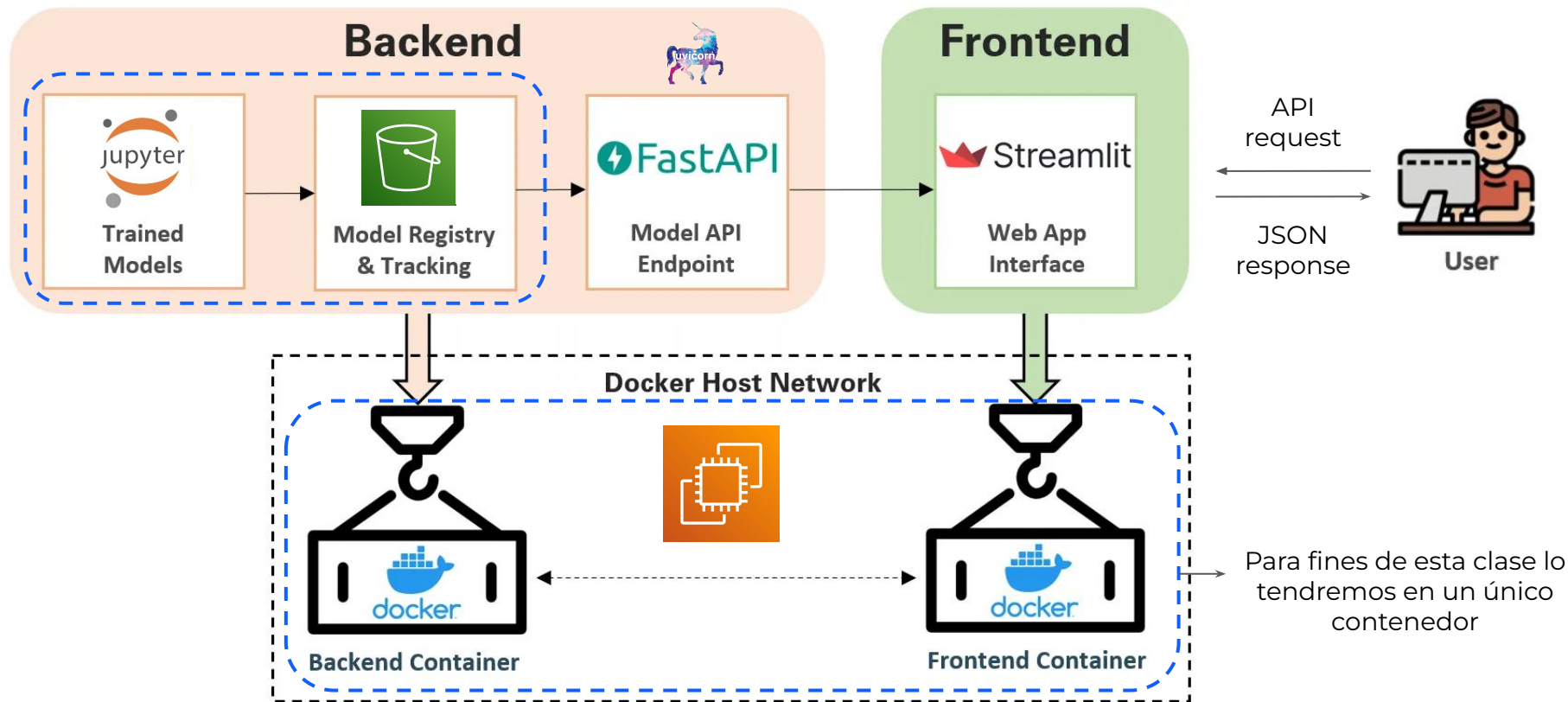
EC2 permite a los usuarios lanzar instancias virtuales con diferentes configuraciones de hardware y software pudiendo **escalar verticalmente y horizontalmente**.

Además las instancias EC2 son altamente configurables y proporcionan varias opciones de **almacenamiento, red y seguridad**, pudiéndose utilizar con distintos propósitos, como hosting de sitios web, procesamiento de datos, aplicaciones empresariales y más.



¿Cómo juntamos las piezas anteriores?

marvik.



marvik.

Python

Python es el lenguaje elegido para IA y despliegue moderno.

Su simplicidad, comunidad activa y amplia disponibilidad de librerías lo hacen ideal para proyectos de NLP, desarrollo web, APIs, y análisis de datos.

Ventajas clave:

- Sintaxis legible y elegante → ideal para principiantes y profesionales.
- Gran ecosistema: pandas, scikit-learn, transformers, numpy, matplotlib, etc.
- Uso transversal: notebooks, desarrollo backend, dashboards, pipelines de datos.
- Compatible con frameworks modernos: FastAPI, Streamlit, PyTorch, HuggingFace.

En este curso usamos Python para entrenar modelos, desplegar APIs y construir frontends interactivos. Aprendemos a organizar código, manejar dependencias, y migrar notebooks a producción.

Organización del proyecto

- Separar código por responsabilidad: api/, utils/, models/, services/
- Usar `__init__.py` para definir paquetes y facilitar importaciones
- Crear un archivo `main.py` o `app.py` como punto de entrada

Manejo profesional de dependencias

- `requirements.txt` → lista simple de paquetes
- `poetry` → control avanzado con `pyproject.toml`, ideal para proyectos compartidos

Testing y logging

- Usar `pytest` para testear funciones críticas
- Incorporar logs con logging para trazabilidad y debugging

Recomendaciones generales

- Escribir funciones reutilizables, breves y bien documentadas
- Evitar hardcodear rutas o credenciales: usá variables de entorno (`dotenv`, `os.environ`)
- Tippear tus funciones con `typing` para mayor claridad y autocompletado

Clonación de repositorio y set up

- **Instalar git** (si no está instalado)

```
sudo apt install git -y  
git --version
```

- **Clonar repositorio** (público)

```
git clone https://github.com/marvik-ai/UM-master-ciencia-de-datos.git  
cd UM-master-ciencia-de-datos
```

En caso de ser repositorio privado se pueden usar algunos métodos como:

- Método 1: Usar SSH Keys
- Método 2: Usar un Personal Access Token (PAT)

- Actualizar paquetes del sistema

```
sudo apt update  
sudo apt upgrade -y
```

- Instalar python (si no viene instalado) y pip

```
sudo apt install python3  
sudo apt install python3-pip -y
```

- Instalar e iniciar docker

```
sudo apt install docker.io -y  
sudo systemctl start docker  
sudo systemctl enable docker
```

- Verificar versiones

```
python3 --version  
pip3 --version  
sudo docker --version
```

Requirements.txt

El archivo requirements.txt es un **estándar** ampliamente utilizado en proyectos de Python para **especificar las dependencias** del proyecto. Contiene una lista de **paquetes** y sus **versiones** que deben ser instalados. Se utiliza junto con pip para instalar todas las dependencias de manera sencilla.

Poetry

Poetry es una **herramienta de gestión de dependencias y empaquetado para proyectos** de Python que busca mejorar y simplificar el manejo de proyectos y sus dependencias. A diferencia de requirements.txt, Poetry utiliza un archivo **pyproject.toml** para definir las dependencias y la configuración del proyecto.



Los entornos virtuales son herramientas que permiten mantener separadas las dependencias requeridas por diferentes proyectos al crear un entorno aislado para cada uno de ellos. Esto significa que cada proyecto puede tener sus propias dependencias, independientemente de las dependencias de otros proyectos, lo cual evita conflictos entre versiones de paquetes

A continuación veamos los comandos para su instalación y creación para el proyecto:

- Instalar venv

```
sudo apt install python3.12-venv -y
```

- Crear entorno virtual por script (incluye instalación de dependencias)

```
chmod +x setup_api.sh  
./setup_api.sh
```

El script incluye

```
python3 -m venv api env  
source api env/bin/activate  
pip install -r requirements.txt
```

- Instalar dependencias con requirements (sin script)

```
pip install -r requirements.txt
```

- Instalar dependencias con poetry

```
poetry install
```

En caso de ir por poetry se debe instalar

```
# Instalar Poetry
```

```
curl -sSL https://install.python-poetry.org | python3 -
```

```
# Agregar Poetry al PATH
```

```
echo 'export PATH="$HOME/.local/bin:$PATH"' >> ~/.bashrc
```

```
source ~/.bashrc # o source ~/.zshrc
```

```
# Verificar la instalación de Poetry
```

```
poetry --version
```


Pruebas a través de API y FE

Correr API y para realizar pruebas

marvik.

- Ejecutar API en una screen (activar entorno previamente)

```
screen  
source api env/bin/activate  
uvicorn app:app --host 0.0.0.0 --port 8000 &
```

- Ejecutar Streamlit en otra screen (activar entorno previamente)

```
Ctrl-a seguido de d (para regresar de la screen anterior)  
screen  
source api env/bin/activate  
streamlit run front_streamlit.py --server.port=8501  
--server.headless=True &
```

Terminamos los procesos de cualquier terminal con `Ctrl-c`

Realizamos pruebas mediante:

- API Request con Postman o Thunder Client
- Frontend



Postman

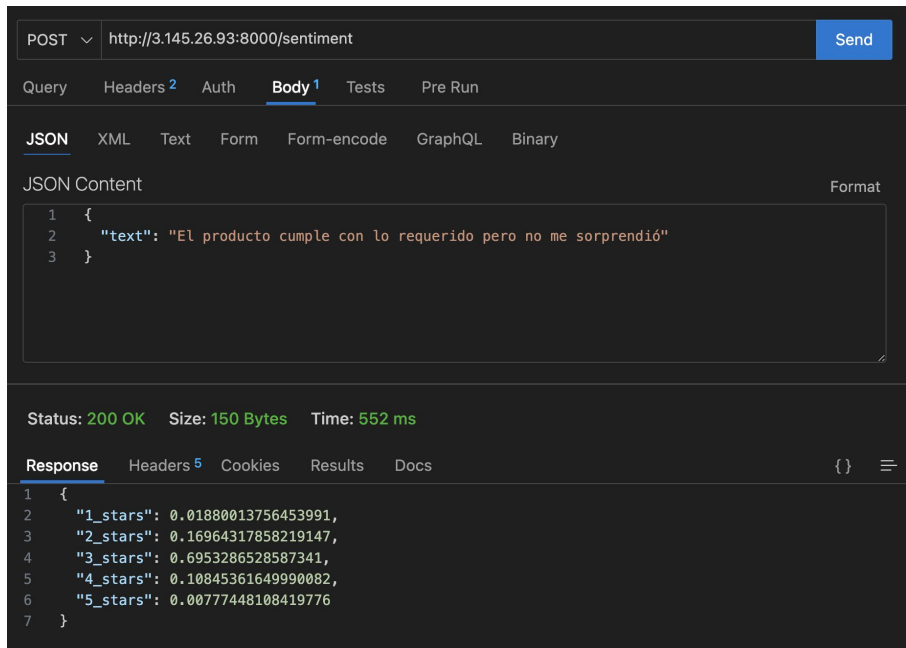


Thunder



Front

Realizar request tipo POST hacía <https://localhost:8000> utilizando Thunder (VsCode) o Postman. Tipo JSON dentro de Body, con único campo 'text' y en value la review.



Ingresa en el explorador a: <https://localhost:8501>

Análisis de sentimiento de reviews

Escribe una review y obtén una calificación de "★" a "★★★★★" estrellas:

Escribe tu review aquí:

Excelent!

Analizar review

Calificación de review: ★★★★★ (5 de 5 estrellas)

marvik.

Despliegue con Docker

Dockerfile

Un Dockerfile es un archivo de texto con una serie de **instrucciones** que se utilizan para **crear** una **imagen** Docker. Cada instrucción en un Dockerfile ejecuta un comando en el contexto de una imagen base y contribuye a construir la nueva imagen **capa por capa**.

A continuación los comandos que debemos utilizar:

- Crear imagen de docker (no olvidar el punto "." del comando)

```
sudo docker build -t sentiment-analysis-app .
```

- Ejecutar imagen de docker (desde una screen)

```
sudo docker run -p 8000:8000 -p 8501:8501 sentiment-analysis-app
```

- Ejecuta un contenedor Docker

```
docker run [OPTIONS] IMAGE [COMMAND] [ARG...]
```

- Lista los contenedores en ejecución

```
docker ps [OPTIONS]
```

- Lista todos contenedores (incluso los detenidos)

```
docker ps -a
```

- Detiene uno o más contenedores en ejecución

```
docker stop [OPTIONS] CONTAINER [CONTAINER...]
```

- Inicia uno o más contenedores detenidos

```
docker start [OPTIONS] CONTAINER [CONTAINER...]
```

- Elimina uno o más contenedores

```
docker rm [OPTIONS] CONTAINER [CONTAINER...]
```

Docker Run

El comando docker run se utiliza para **crear un nuevo contenedor** a partir de una imagen especificada. Inicia el contenedor ejecutando el comando especificado en el CMD o el **ENTRYPOINT** del Dockerfile, o cualquier comando que se especifique directamente en la línea de comando.

Docker Start

El comando docker start se usa para **iniciar un contenedor existente** que ha sido previamente **detenido**. Simplemente reinicia el proceso principal en un contenedor que ya fue creado y detenido. **No modifica ninguna configuración del contenedor**; simplemente lo vuelve a poner en marcha desde su estado detenido.

- Eliminar uno o más contenedores

```
docker rm [OPTIONS] CONTAINER [CONTAINER...]
```

- Listar imágenes disponibles

```
docker images [OPTIONS] [REPOSITORY[:TAG]]
```

- Elimina una o más imágenes

```
docker rmi [OPTIONS] IMAGE [IMAGE...]
```

- Obtiene los logs de un contenedor

```
docker logs [OPTIONS] CONTAINER
```

- Descarga una imagen desde un registro

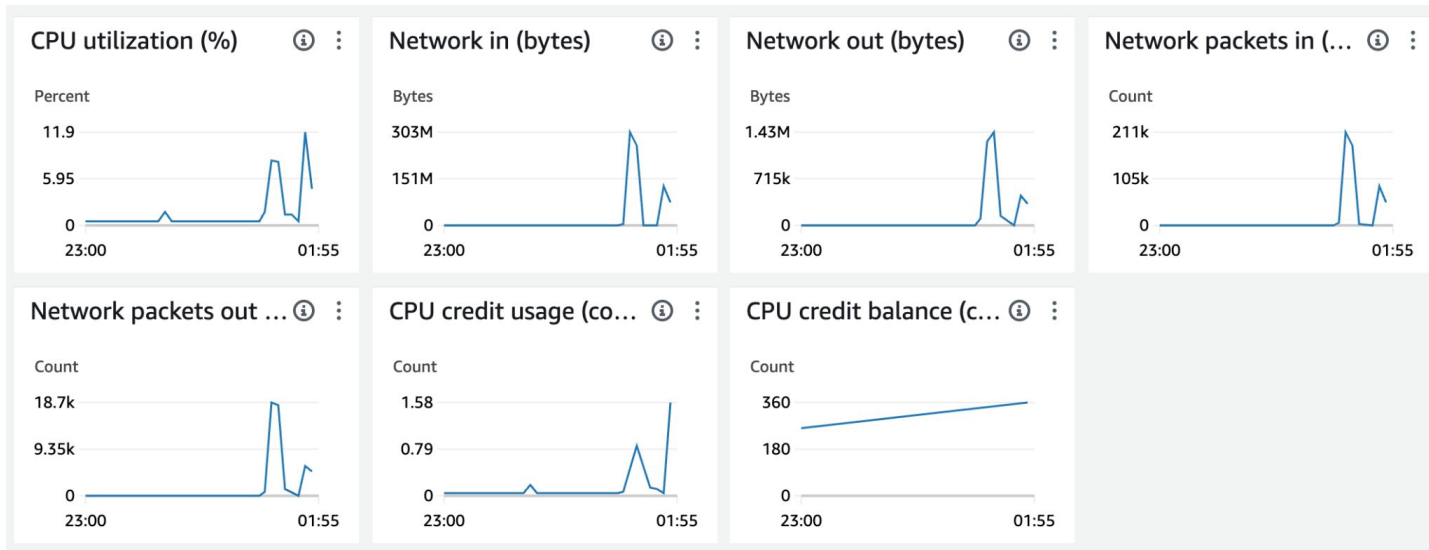
```
docker pull [OPTIONS] NAME[:TAG|@DIGEST]
```

- Ejecuta un comando en un contenedor en ejecución

```
docker exec [OPTIONS] CONTAINER COMMAND [ARG...]
```

Monitoreo y buenas Prácticas en la nube

El monitoreo continuo es vital para mantener la operatividad y eficiencia de la aplicación. Utilizamos herramientas como CloudWatch para monitorear el rendimiento de la aplicación y responder rápidamente a cualquier incidente. Además, la configuración de alertas automáticas nos permite estar proactivamente informados sobre cualquier comportamiento anómalo.



Seleccionar AMI y tipo de instancia adecuada

- Utiliza una AMI que sea compatible con tus necesidades (e.g., Ubuntu 20.04 LTS).
- Asegúrate de que la instancia tenga los recursos necesarios (CPU, RAM, almacenamiento) para manejar la carga esperada.

Seguridad

- Configura grupos de seguridad de manera adecuada para permitir solo el tráfico necesario (SSH, HTTP, HTTPS).
- Usa claves SSH para acceder de manera segura a la instancia.
- Considera usar IAM roles para gestionar permisos de acceso a otros servicios de AWS.

Python y dependencias

- Usa entornos virtuales Python y evitar conflictos de dependencias.
- Usa archivos como requirements o pyproject para manejar dependencias de manera eficiente.
- Actualiza y verifica regularmente las dependencias para mantener la seguridad y estabilidad.

Optimización de Dockerfile

- Mantén el Dockerfile limpio y optimizado, eliminando capas innecesarias.
- Usa imágenes base ligeras como `python:3.8-slim` para reducir el tamaño de la imagen.

Seguridad en Docker

- No incluyas credenciales sensibles en el Dockerfile.
- Usa variables de entorno para manejar configuraciones sensibles.
- Escanea (AWS ECR o Docker Scan) las imágenes Docker regularmente en busca de vulnerabilidades.

Etiquetado y Versionado

- Etiqueta las imágenes Docker con números de versión para un despliegue consistente.
- Mantén un registro de los cambios y versiones de las imágenes.

Automatización del Despliegue

- Automatiza el despliegue con CI/CD pipelines para garantizar proceso repetible y consistente.

Monitorización y Logs

- Implementa soluciones de monitorización como CloudWatch para AWS.
- Configura logs adecuados para rastrear errores y rendimiento de la aplicación.

Escalabilidad

- Diseña la arquitectura de manera que sea fácil escalar horizontalmente.
- Considera el uso de balanceadores de carga para distribuir el tráfico.

Mantenimiento y Actualización

- Planifica el mantenimiento regular de la instancia y los contenedores.
- Mantén el sistema operativo y el software actualizados para asegurar la seguridad y el rendimiento.

Manejo de Datos

- Asegúrate de cumplir con las regulaciones de privacidad y protección de datos.
- Usa cifrado en tránsito y en reposo para datos sensibles.

Pruebas

- Realiza pruebas exhaustivas de tu API y frontend antes del despliegue.
- Implementa pruebas automatizadas para verificar la funcionalidad de la API y el frontend.

¡Gracias!

¿Preguntas?
santiago@marvik.ai