Informe de Proyecto DataOps – Diplomado de Ingeniería de Datos

# 1. Introducción

Este proyecto consiste en el desarrollo de un pipeline ETL automatizado utilizando principios de DataOps. Se integraron datos desde una base de datos PostgreSQL, se aplicaron transformaciones con Python, y el proceso fue empaquetado usando Docker. Finalmente, se implementa una etapa de CI/CD mediante Jenkins para permitir despliegues controlados.

# 2. Estructura del Proyecto

La estructura del proyecto está organizada en carpetas de la siguiente forma:

dataops-rrhh/  
├── etl/  
│ ├── main.py  
│ ├── requirements.txt  
├── output/  
│ └── empleados\_transformados\_<timestamp>.csv  
├── Dockerfile  
├── jenkins/  
│ └── Jenkinsfile (por implementar)  
├── README.md

# 3. Entorno de Desarrollo

Se utilizó Python 3.10 en un entorno virtual, con las siguientes librerías principales: pandas, sqlalchemy, psycopg2-binary. Las dependencias están listadas en requirements.txt. Se utilizó VSCode como entorno de desarrollo.

Además del script principal main.py, se trabajó inicialmente en un archivo main.ipynb (Jupyter Notebook), el cual fue utilizado para pruebas exploratorias, conexión a la base de datos y validación de transformaciones antes de consolidar el pipeline final. Por esta razón, el archivo requirements.txt incluye la librería notebook.

# 4. Desarrollo del ETL

El script main.py realiza la conexión a PostgreSQL, lee los datos de la tabla rrhh.empleado, aplica transformaciones como conversión de texto, normalización, creación de campos derivados y exportación a un archivo CSV con timestamp.

Conexión a la base de datos PostgreSQL

Texto

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.

Ejecución de transformaciones: Texto

El contenido generado por IA puede ser incorrecto.

Disposición de archivo CSV:

Texto

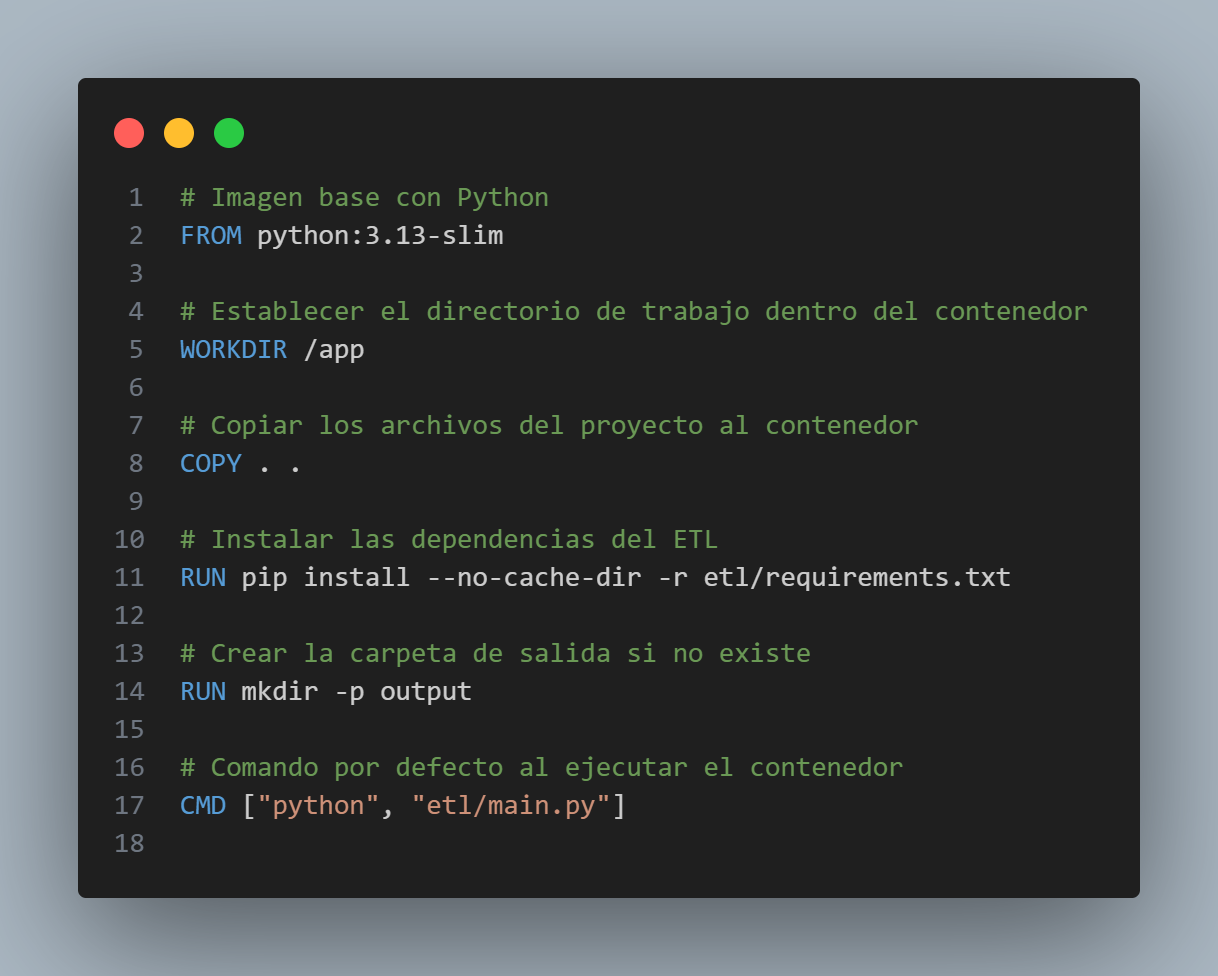
El contenido generado por IA puede ser incorrecto.

# 5. Carga de Datos Transformados

El archivo resultante se exporta en formato CSV a la carpeta output/. El nombre del archivo incluye la fecha y hora de ejecución como medida de control y trazabilidad.

# 6. Empaquetado con Docker

Se creó un Dockerfile que define una imagen ligera basada en python:3.10-slim. La imagen contiene el código del ETL, las dependencias y un comando CMD para ejecutar el script automáticamente al correr el contenedor.



*Nota: durante el desarrollo se presentaron dificultades al montar volúmenes usando Docker desde Git Bash en Windows debido a rutas con espacios. Se solucionó exitosamente utilizando winpty para ejecutar el contenedor, permitiendo que las rutas fueran interpretadas correctamente por Docker Desktop.*

Captura de pantalla de computadora

El contenido generado por IA puede ser incorrecto.

# 7. Automatización con Jenkins (CI/CD)

Automatización del entorno Jenkins

Para este proyecto se utilizó una máquina virtual con sistema operativo Linux (Ubuntu) en la que se instaló Docker, Java y Git utilizando un script automatizado llamado instalacionDocker.sh. Este archivo contiene todos los comandos necesarios para configurar el entorno base de trabajo, permitiendo dejar la VM lista sin necesidad de instalaciones manuales.

Una vez completada la instalación, Jenkins se desplegó como contenedor utilizando el archivo docker-compose.yml. Este archivo levanta el servicio en el puerto 80, accediendo a Jenkins desde el navegador. Desde la interfaz web de Jenkins se creó manualmente un pipeline, replicando el procedimiento visto en clase.

El pipeline fue configurado para realizar los siguientes pasos:

Clonar el repositorio con el código fuente del proyecto

Construir la imagen Docker con el nombre etl-job

Ejecutar el contenedor, el cual corre el script ETL en Python y genera como salida un archivo .csv en la carpeta output

Este enfoque demuestra una implementación de CI/CD real sin necesidad de un Jenkinsfile, aprovechando las capacidades de configuración manual desde la consola web de Jenkins.

# 8. Conclusiones

Este proyecto permitió aplicar principios reales de DataOps, desde la conexión a una fuente de datos relacional hasta la ejecución automática vía contenedores. Se reforzó el uso de versionamiento, entornos aislados y automatización. Futuros pasos podrían incluir validaciones de datos, alertas, y pruebas automatizadas.