

Resolución de Desafío Práctico para Desarrollador de Datos Geoespaciales

Introducción:

El presente documento tiene como objetivo principal exponer la metodología, el desarrollo y los resultados obtenidos en respuesta al desafío práctico planteado por Garruchos Agropecuaria para la posición de Desarrollador de Datos Geoespaciales.

Diseño y formato de entrega:

La solución se estructura en un **repositorio de GitHub** que incluye:

- Código Fuente Completo: Scripts en Python, con comentarios detallados.
- Archivo README.md: Explicación para la ejecución del código.
- Archivo requirements.txt: donde se detallan las librerías necesarias
- Datasets: brindados por empresa
- Outputs: .CSV, .GeoPackage y QGIS Project
- Docker: archivos abajo listados para crear el contenedor
- Este Informe Técnico: detallando la justificación de las decisiones metodológicas y los resultados clave.

PARTE 1: El objetivo de esta primera parte fue crear un entorno reproducible que permita trabajar con datos geoespaciales utilizando **PostgreSQL** (versión ≥ 14), **PostGIS** (versión ≥ 3) y opcionalmente la extensión **h3-pg**, dentro de un contenedor Docker.

1. Creación del entorno Docker

El proyecto se construyó bajo el principio de Contenerización para garantizar la reproducibilidad y el aislamiento de dependencias. La infraestructura se levantó utilizando *Docker Desktop* en mi máquina local (*Windows*) para gestionar contenedores y la orquestación mediante *Docker Compose*.

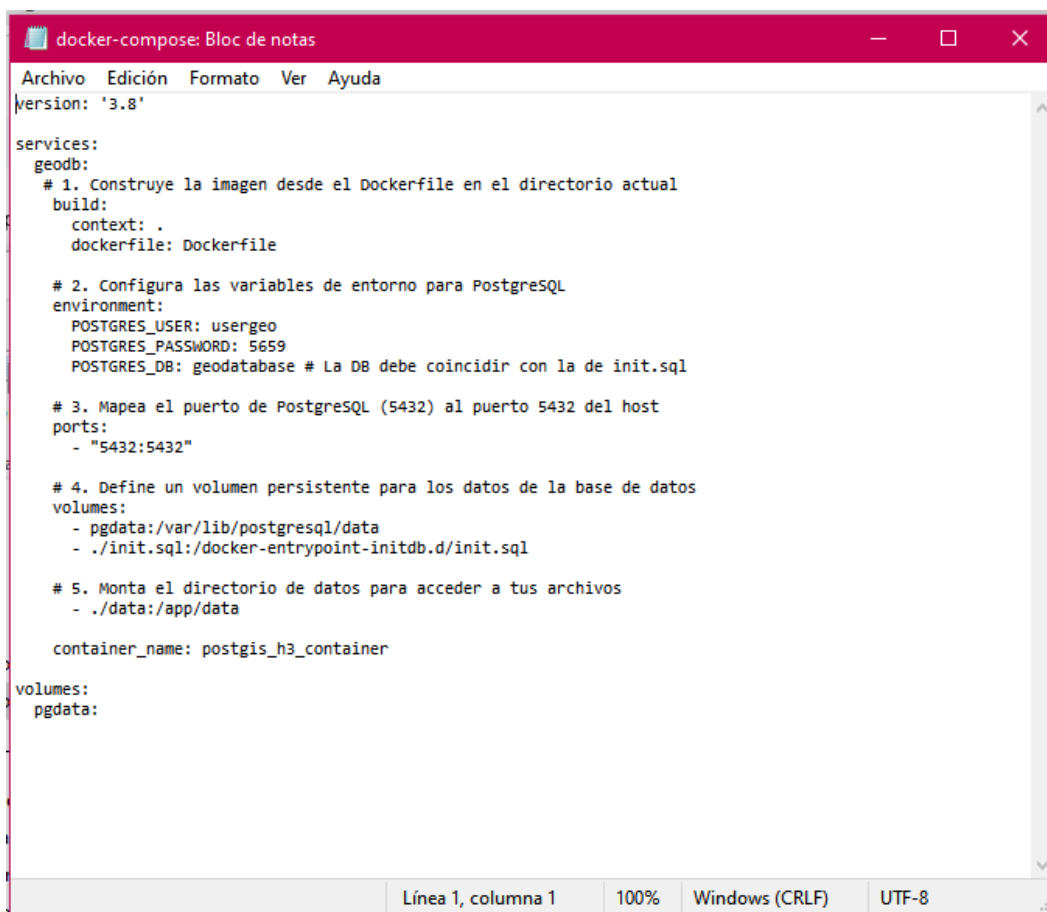
Primero se construyó un **Dockerfile** basado en la imagen oficial de PostgreSQL 14. Durante el proceso de construcción se instalaron los paquetes necesarios para habilitar **PostGIS**, asegurando compatibilidad con datos espaciales vectoriales y ráster.

El Dockerfile garantiza que el contenedor resultante disponga de un entorno completamente funcional para almacenar, consultar y analizar datos espaciales desde diferentes fuentes.

Luego se configuró un archivo **docker-compose.yml** que define y administra el servicio de base de datos.

En este archivo se establecieron:

- Variables de entorno para crear automáticamente la base (`POSTGRES_DB`, `POSTGRES_USER`, `POSTGRES_PASSWORD`).
- Exposición del puerto `5432` para conectar desde QGIS o DBeaver.
- Un **volumen persistente** (`pgdata`) para conservar los datos incluso si el contenedor se detiene o elimina.
- El montaje del script `init.sql`, encargado de crear las extensiones necesarias al inicializar la base.

A screenshot of a text editor window titled "docker-compose: Bloc de notas". The window displays a Docker Compose configuration file for a PostgreSQL database. The configuration includes a version, a service named "geodb" with build instructions, environment variables for user, password, and database name, port mapping for 5432, volume definitions for "pgdata" and "init.sql", and container name "postgis_h3_container".

```
Archivo Edición Formato Ver Ayuda
version: '3.8'

services:
  geodb:
    # 1. Construye la imagen desde el Dockerfile en el directorio actual
    build:
      context: .
      dockerfile: Dockerfile

    # 2. Configura las variables de entorno para PostgreSQL
    environment:
      POSTGRES_USER: usergeo
      POSTGRES_PASSWORD: 5659
      POSTGRES_DB: geodatabase # La DB debe coincidir con la de init.sql

    # 3. Mapea el puerto de PostgreSQL (5432) al puerto 5432 del host
    ports:
      - "5432:5432"

    # 4. Define un volumen persistente para los datos de la base de datos
    volumes:
      - pgdata:/var/lib/postgresql/data
      - ./init.sql:/docker-entrypoint-initdb.d/init.sql

    # 5. Monta el directorio de datos para acceder a tus archivos
      - ./data:/app/data

    container_name: postgis_h3_container

volumes:
  pgdata:
```

Gracias a `docker-compose`, la base de datos se puede levantar o detener fácilmente sin perder configuraciones ni datos.

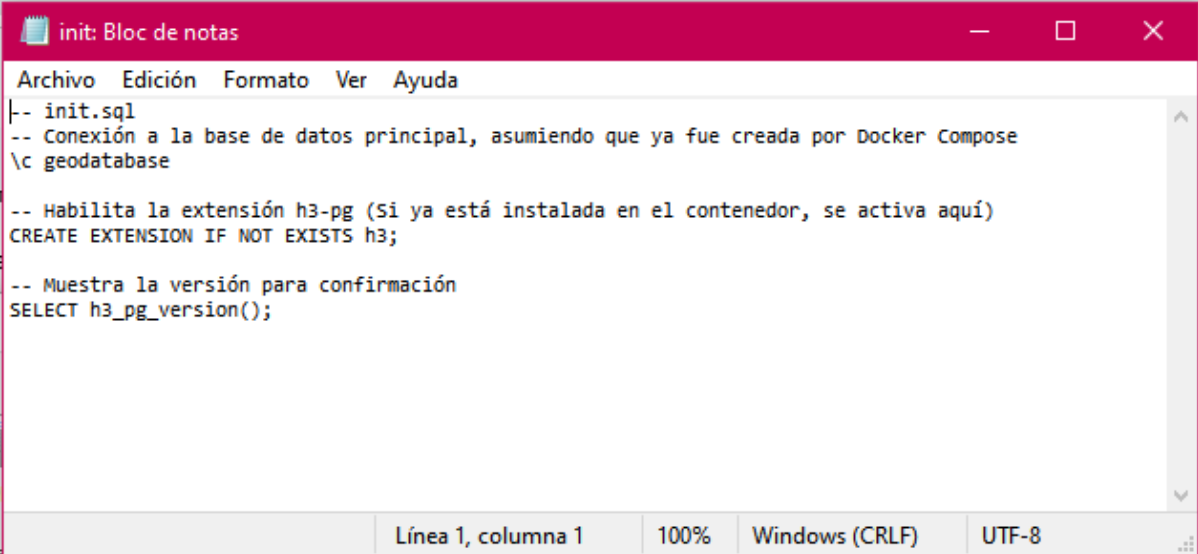
Este entorno servirá como base para cargar los datasets del lote L4 "La Magdalena" (KML, GeoParquet y GeoPackage) y realizar los análisis espaciales posteriores.

Además, se incorporó un script de inicialización que se ejecuta automáticamente la primera vez que se crea la base de datos.

Este script:

- Conecta a la base definida por Docker Compose.
- Crea la extensión `h3-pg` (si fue instalada en la imagen).
- Ejecuta una consulta para verificar que la extensión se haya habilitado correctamente.

El script se monta dentro del contenedor en la ruta `/docker-entrypoint-initdb.d/init.sql`, que es el directorio desde el cual PostgreSQL ejecuta automáticamente los scripts de inicialización.



```
init: Bloc de notas
Archivo  Edición  Formato  Ver  Ayuda
|-- init.sql
-- Conexión a la base de datos principal, asumiendo que ya fue creada por Docker Compose
\c geodatabase

-- Habilita la extensión h3-pg (Si ya está instalada en el contenedor, se activa aquí)
CREATE EXTENSION IF NOT EXISTS h3;

-- Muestra la versión para confirmación
SELECT h3_pg_version();

Línea 1, columna 1    100%    Windows (CRLF)    UTF-8
```

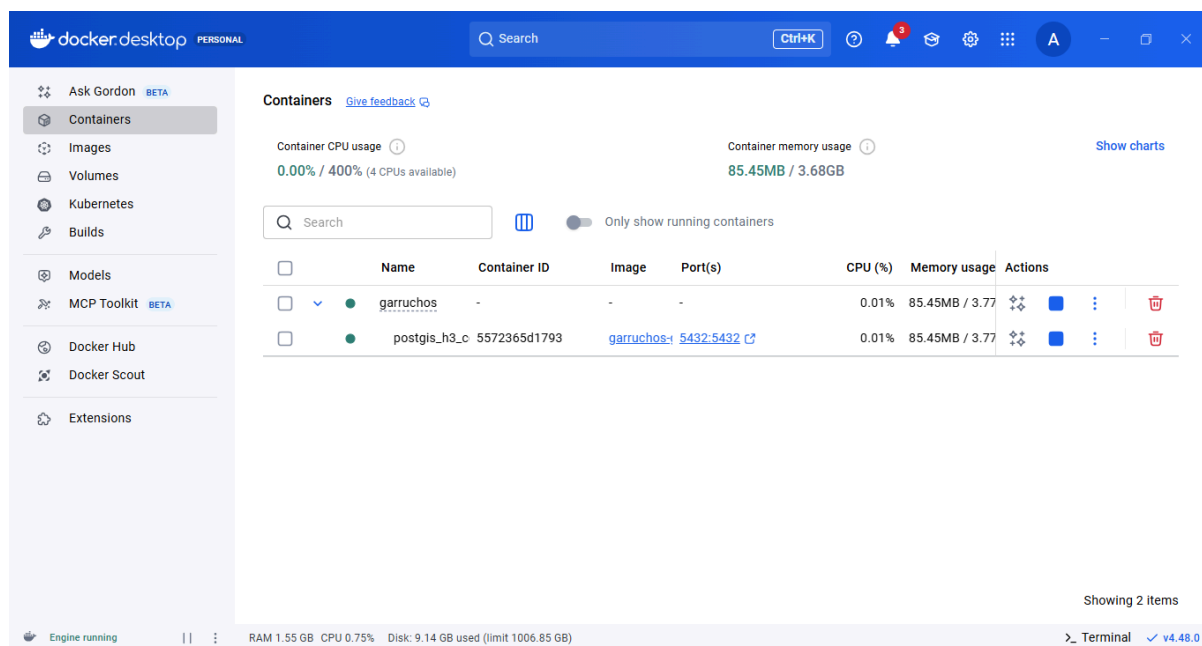
2. Levantamiento inicial del entorno

Esto construye la imagen con PostGIS, crea el contenedor, inicializa la base de datos y ejecuta automáticamente el `init.sql`.

```
./ docker-compose build
```

```
./ docker-compose up -d
```

```
./ docker ps → verifica que el contenedor esté activo
```



Captura de la página principal de Docker Desktop donde se observa el contenedor activo.

3. Carga inicial de datos

Una vez que el contenedor estuvo activo, utilicé la herramienta de línea de comandos ogr2ogr (instalada dentro del contenedor como parte de gdal-bin) para cargar los datos:

- Se cargó con éxito el archivo KML: 'la magdalena L4.kml'
- ```
docker exec postgis_h3_container ogr2ogr -f "PostgreSQL"
"PG:host=localhost user=usergeo password=5659 dbname=geodatabase"
/app/data/la_magdalena_L4.kml -ln kml_layer -overwrite
```
- Se cargó con éxito el archivo GeoPackage: 'veris\_data.gpkg'
- ```
docker exec postgis_h3_container ogr2ogr -f "PostgreSQL"
"PG:host=localhost user=usergeo password=5659 dbname=geodatabase"
/app/data/veris_data.gpkg -ln geopackage_layer -overwrite
```

3.1. Resolución para la carga del archivo formato GeoParquet

El archivo GeoParquet (soy_performance_2019_2021_2023.parquet) presentó un desafío de incompatibilidad, lo que me obligó a cambiar la estrategia de carga a un enfoque programático.

El binario de ogr2ogr instalado en el contenedor no incluía el driver necesario para leer el formato GeoParquet. Decidí usar un entorno Python externo, aprovechando que el contenedor de Docker estaba accesible en localhost:5432.

Para evitar fallos en la instalación de dependencias binarias críticas en Windows (como GDAL y Fiona) que son comunes con pip, utilicé Miniconda para crear un entorno aislado (geonv) e instalar de forma estable las librerías GeoPandas, Shapely, y GeoAlchemy2. El archivo es cargar_geoparquet.py,

El orden para los comandos fue:

-conda init

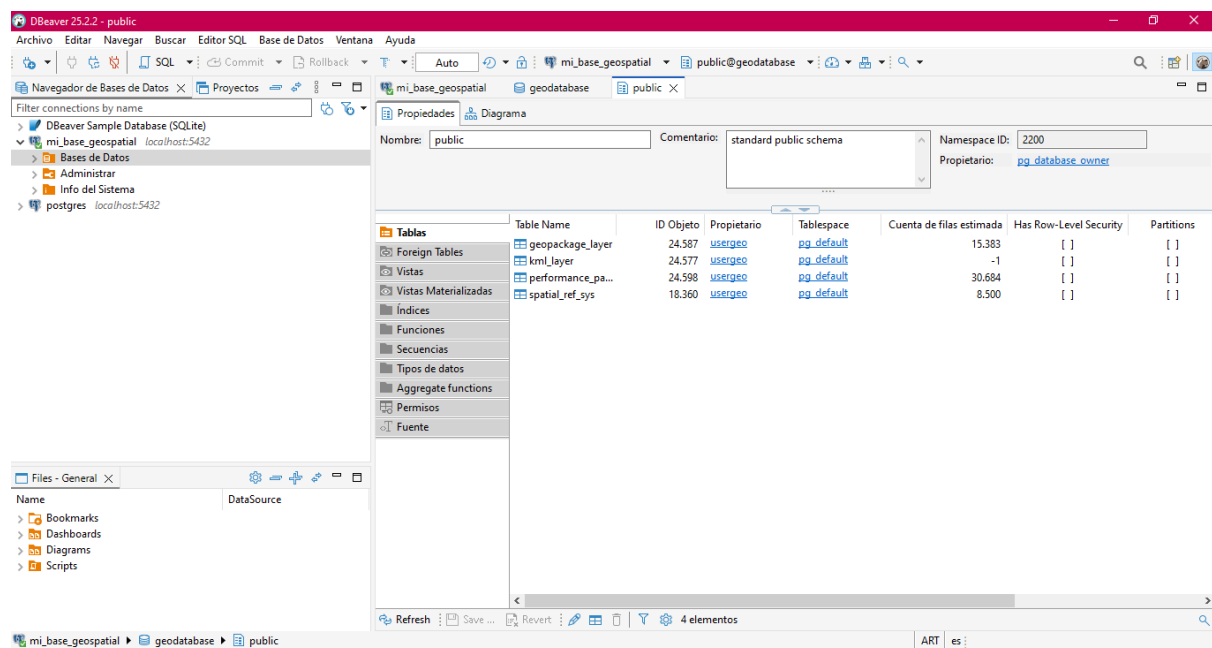
-conda create -n geonv python=3.11

-conda activate geonv

-conda install geopandas sqlalchemy psycopg2 fastparquet pyarrow geoalchemy2

-python cargar_geoparquet.py

Una vez cargados exitosamente los datos geoespaciales, se chequea en el entorno DBeaver como se observa en la siguiente imagen. Dentro de la Base de Datos, en la solapa Tablas, se encuentran las capas cargadas.



Chequeo del estado: por ejemplo se eligió la capa de datos del sensor veris → la tabla en DBeaver tiene este aspecto:

<

Adicionalmente, se utilizó el visor espacial integrado de DBeaver para visualizar las capas y validar la correcta ubicación de los datos en el mapa.

Por otra parte, se comparte en GitHub un archivo qgz, este archivo tiene las capas cargadas tanto de entrada como los outputs. Permite la visualización directa y un análisis preliminar de los resultados.

Si bien la infraestructura Docker/PostGIS fue configurada con éxito y los datos geospaciales se cargaron correctamente (tal como se evidencia en el Dockerfile y el script de carga), la integración bidireccional entre Google Colab y el contenedor local presentó desafíos de red y seguridad. Para asegurar la finalización exitosa del desafío dentro del plazo, se optó por una estrategia de análisis enfocada en Python, manteniendo la base de datos Docker como el ambiente de producción ideal para una implementación a largo plazo.

Parte 2. Análisis de datos de rendimiento del cultivo. (Python)

Todas las variables (rendimiento, conductividad eléctrica y altimetría) se expresan como valores relativos (%) respecto al rango del lote, lo que permite comparar su variabilidad espacial interna.

Parte 2.1 – Evaluación del rendimiento y correlación con propiedades del suelo.

En esta primera parte se integraron y analizaron los datos de rendimiento de soja correspondientes a las campañas 2019, 2021 y 2023, junto con la información de conductividad eléctrica superficial y subsuperficial y altimetría relativas obtenidas del sensor Veris en el Lote 4 - La Magdalena (Córdoba, Argentina).

Se utilizó el entorno de Google Colaboratory (Colab) para el desarrollo del *script* en Python. Se emplearon herramientas como GeoPandas, Pandas, NumPy, Matplotlib, Seaborn, H3, PySAL y Scikit-learn para el procesamiento, análisis espacial y visualización de los datos.

En principio se hizo un análisis exploratorio de los archivos de entrada para identificación de valores nulos y/o outliers, y se renombraron columnas para agilizar el manejo de los datos. En esta instancia se decidió unificar los datasets para un mejor manejo espacial. En ese sentido se utilizó 'sjoinnearset' para unir los datos de ambos dataset, por la clave h3_res12 (generada previamente para los datos del sensor Veris). Luego, se llevó a cabo un análisis de correlación de Pearson y Spearman con el objetivo de cuantificar la relación entre las propiedades del suelo y el rendimiento relativos por campaña.

Finalmente, se aplicó aprendizaje no supervisado utilizando la técnica de agrupamiento (K-Means) integrando las tres campañas, con el objetivo de identificar patrones espaciales consistentes en el tiempo. Para encontrar K-óptimo se usó el Método del Codo. Además, se generó y exportó un archivo GeoPackage ('clusters_rendimiento.gpkg') que contiene los clusters de rendimiento relativo obtenidos, con el fin de visualizarlos en QGIS y permitir un análisis espacial más detallado. La tabla de valores medios por cluster, que funciona como el resumen estadístico de cada zona (Rendimiento, CE subsup, CE sup y Altimetría), se exportó a formato CSV como 'analisis_zonas.csv'. Dado que las variables están expresadas en forma relativa, el enfoque de clustering permite detectar zonas con comportamiento productivo estable —independientemente de las variaciones interanuales— y evaluar la influencia conjunta de las propiedades del suelo sobre la productividad.

Podrían explorarse modelos no lineales (por ejemplo, Random Forest o regresión polinómica) en futuras etapas.

Parte 2.2 – Identificación de hotspots y coldspots de rendimiento en la campaña 2021.

En esta segunda parte de la prueba, se filtró el dataset por año de campaña (2021) y se aplicó el estadístico Getis-Ord G_i^* utilizando la librería PySAL para identificar zonas calientes (hotspots) de rendimiento relativo de soja. El análisis se aplicó sobre las celdas H3 generadas en la Parte 2.1, utilizando la matriz de pesos espaciales de primer orden (Queen contiguity) para evaluar la autocorrelación local. Se realizó con 95% de confianza, $p < 0.05$. Este análisis permitió detectar agrupamientos espaciales significativos de valores altos, aunque no se identificaron coldspots estadísticamente significativos. Se observaron principalmente sectores con rendimientos superiores al promedio, mientras que las zonas de bajo rendimiento no presentaron una autocorrelación espacial fuerte. No obstante, al comparar estos resultados con el análisis de K-Means realizado en la Parte 2.1, se reconoce que el cluster 1, coincide casi en su totalidad con las áreas identificadas como hotspots. Esta coincidencia valida de forma cruzada el análisis no supervisado, demostrando que el cluster 1 es la zona más estable y consistente de alto rendimiento relativo del lote.

Por último, se generó y exportó un archivo GeoPackage ('Analisis_Hotspot_Rendimiento_2021.gpkg') con los resultados del análisis de hotspots, diseñado para su visualización en QGIS, lo que permitirá superponer ambos mapas (clusters y hotspots) y caracterizar con mayor precisión las zonas del lote que presentan diferencias significativas de productividad. Este enfoque aporta información clave para el diseño de estrategias de manejo sitio-específico orientadas a mejorar la eficiencia y la productividad.

Parte 2.3. Análisis de baja productividad.

¿Cuál podría ser la razón de la baja productividad en las zonas con peor rendimiento?

Los resultados integrados de las tres campañas muestran que las zonas de bajo rendimiento relativo se asocian con mayores valores de conductividad eléctrica y menor altimetría relativas. Dado que todas las variables se expresan en términos porcentuales, estas relaciones reflejan tendencias espaciales internas del lote, no valores absolutos. En conjunto, los patrones observados sugieren que la variabilidad productiva está condicionada por diferencias locales en las propiedades del suelo —posiblemente vinculadas a textura, humedad o acumulación de sales—, lo que resalta la utilidad del análisis espacial para definir estrategias de manejo sitio-específico orientadas a optimizar la productividad.

¿Podrías sugerir acciones conjuntas con el equipo de campo para mejorar el rendimiento en esas áreas?

Se sugiere trabajar junto al equipo de campo en la validación de las zonas de bajo rendimiento relativo, mediante observaciones y muestreos dirigidos que permitan identificar causas edáficas o de manejo. Con esa información, podrían aplicarse intervenciones sitio-específicas —como ajustes de drenaje, nutrición o densidad de siembra— orientadas a mejorar el rendimiento y reducir la variabilidad interna del lote.

Preguntas finales:

Si tuvieras que integrar datos de rendimiento, suelo, satélite y clima de forma continua, ¿qué pasos o herramientas utilizarías para garantizar la calidad, la coherencia y la trazabilidad de los datos a lo largo del tiempo?

Para integrar datos de forma continua (rendimiento, suelo, satélite y clima), me centraría en un flujo de trabajo de ELT (Extraer, Cargar, Transformar):

Calidad y Coherencia: Usaría Python/Pandas para realizar una validación de esquema y dominio al inicio de la ingesta (validando tipos de datos y rangos de valores) y estandarizaría todas las geometrías al CRS EPSG:4326 en PostgreSQL/PostGI. Cualquier dato que falle la validación se marca y aísla.

Trazabilidad: Aprovecharía las capacidades de PostgreSQL/PostGIS. Cada tabla de datos transformados incluiría columnas de metadatos como `source_file` (nombre del archivo de origen), `processing_timestamp` (momento de la carga), y `code_version` (un tag de Git).

Herramientas Clave: Usar Python/Pandas para la transformación y validación inicial, y PostgreSQL/PostGIS como la base de datos central robusta para almacenar, indexar y consultar todos los datos geoespaciales.

¿Cómo adaptarías tu flujo de trabajo actual para que pueda ejecutarse en muchos campos y campañas mientras sigue siendo reproducible y mantenible?

Para escalar el proceso a muchos campos y campañas, va a ser crucial hacer que el pipeline sea genérico y modular.

Reproducibilidad con Docker: La principal acción sería contenedorizar el entorno de procesamiento (todos los scripts de Python/SQL y sus dependencias) utilizando Docker.

Parametrización: Adaptaría los scripts para que sean agnósticos al campo. En lugar de tener una variable codificada como `campo = 'La Magdalena'`, el script aceptaría un parámetro (e.g., `field_id`) al ser ejecutado. Así, el mismo código se puede llamar para 50 campos diferentes en un bucle o una lista de tareas.

Utilizaría una herramienta de orquestación como Apache Airflow (o en su defecto, scripts avanzados de Bash/Python). Con Airflow, es posible automatizar y encadenar los pasos del pipeline de forma robusta.

PROPUESTA DE ABORDAJE DEL EJERCICIO BONUS

Para anticipar el rendimiento durante el ciclo de crecimiento del cultivo, propongo incorporar un índice espectral derivado de imágenes Sentinel-2, como el NDVI (Normalized Difference Vegetation Index), correspondiente a la campaña 2020–2021.

Técnicamente, el proceso implicaría descargar y procesar las imágenes Sentinel-2 en la plataforma Google Earth Engine (GEE), filtrando por fechas y por el área de estudio. Luego, se calcularía el NDVI mediante la expresión $(B8 - B4) / (B8 + B4)$, utilizando las bandas del infrarrojo cercano (B8) y el rojo (B4). Los valores resultantes se reproyectarían a EPSG:4326 y se recortarían al polígono del lote.

Posteriormente, se podrían calcular estadísticos zonales (media de NDVI) por celda H3 (resolución 12), para luego correlacionar el NDVI medio con el rendimiento relativo de la campaña 2020-2021. Finalmente, se generaría un mapa comparativo entre el NDVI y los clusters de rendimiento, permitiendo analizar si las zonas de mayor vigor vegetativo durante el ciclo del cultivo se corresponden con los sectores de mayor rendimiento.

En síntesis, este enfoque permitiría anticipar patrones de rendimiento antes de la cosecha, brindando información valiosa para el monitoreo y la toma de decisiones agronómicas en tiempo real.