**Plan de integración para Whisper Diarizer**

**Objetivo**

Integrar los nuevos archivos y mejoras con la imagen Docker existente de 16GB de manera eficiente, manteniendo la compatibilidad y aprovechando el trabajo ya realizado.

**Paso 1: Verificación y análisis de la imagen existente**

Ejecuta el script verify\_docker\_image.sh para obtener información detallada sobre tu imagen actual.

chmod +x verify\_docker\_image.sh

./verify\_docker\_image.sh

**Paso 2: Opciones de integración**

**Opción A: Actualizar la imagen existente (recomendada si la imagen está bien configurada)**

1. Crear un nuevo Dockerfile que use tu imagen como base:
2. FROM whisper\_diarizer:latest
3. # Copiar los nuevos archivos
4. COPY whisper\_diarizer\_container.py /app/
5. COPY config.yaml /app/
6. # Actualizar punto de entrada si es necesario
7. ENTRYPOINT ["python3", "whisper\_diarizer\_container.py"]
8. Construir una nueva imagen con esta configuración:
9. docker build -t whisper\_diarizer:updated -f Dockerfile.update .

**Opción B: Exportar los modelos de la imagen existente (si quieres una imagen más limpia)**

1. Crea un contenedor temporal para extraer modelos y archivos importantes:
2. docker run --name temp\_whisper whisper\_diarizer bash -c "mkdir -p /export && cp -r /root/.cache/\* /export/ && tar -czf /export/models.tar.gz /export/"
3. docker cp temp\_whisper:/export/models.tar.gz ./models.tar.gz
4. docker rm temp\_whisper
5. Usa el nuevo Dockerfile que hemos creado y añade instrucciones para incluir estos modelos:
6. # En el Dockerfile:
7. COPY models.tar.gz /tmp/
8. RUN mkdir -p /root/.cache && tar -xzf /tmp/models.tar.gz -C /root/.cache && rm /tmp/models.tar.gz

**Paso 3: Probar la nueva configuración**

**Para la opción A (imagen actualizada):**

docker run --gpus all -v "$(pwd)/input:/app/input" -v "$(pwd)/output:/app/output" whisper\_diarizer:updated --input /app/input --output /app/output

**Para la opción B (imagen nueva con modelos exportados):**

docker run --gpus all -v "$(pwd)/input:/app/input" -v "$(pwd)/output:/app/output" whisper\_diarizer:new --input /app/input --output /app/output

**Paso 4: Documentación y script de ayuda**

1. Actualiza el README.md con instrucciones específicas para tu configuración
2. Adapta el script run\_docker.sh para que use la imagen correcta

**Notas importantes**

* Si la imagen existente ya tiene los modelos descargados (muy probable dado su tamaño de 16GB), la Opción A ahorrará mucho tiempo y ancho de banda
* Si la imagen existente tiene problemas o configuraciones incorrectas, la Opción B te permitirá tener una configuración más limpia
* Guarda una copia de seguridad de la imagen original: docker save whisper\_diarizer > whisper\_diarizer\_backup.tar

**Whisper Diarizer Project Summary**

**Proyecto Actual**

El proyecto es un script de diarización de audio que utiliza:

* Whisper (OpenAI) para transcripción
* Pyannote Audio para identificación de speakers
* Clustering para separar voces

**Archivos Esenciales a Mantener**

1. whisper\_diarizer.py: Script principal de procesamiento
2. Dockerfile: Configuración para contenerización
3. requirements.txt: Dependencias del proyecto
4. .gitignore: Para ignorar archivos innecesarios
5. README.md: Documentación del proyecto

**Archivos para Eliminar**

* Archivos de sistema: .DS\_Store
* Cachés de Python: \_\_pycache\_\_
* Logs temporales
* Archivos de entorno virtual: venv/, .env

**Contenido Recomendado para GitHub**

**README.md**

markdown

Copy

# Whisper Diarizer

## Descripción

Script de diarización de audio que transcribe y separa voces usando Whisper y Pyannote.

## Requisitos

- Python 3.8+

- CUDA (opcional pero recomendado)

- Dependencias en requirements.txt

## Instalación

\`\`\`bash

pip install -r requirements.txt

\`\`\`

## Uso

\`\`\`python

python whisper\_diarizer.py

\`\`\`

## Dockerización

\`\`\`bash

docker build -t whisper-diarizer .

docker run whisper-diarizer

\`\`\`

**.gitignore**

Copy

# Entornos virtuales

venv/

.env

\_\_pycache\_\_/

# Archivos de sistema

.DS\_Store

.vscode/

.idea/

# Archivos de audio y transcripciones

\*.wav

\*.mp3

transcript.txt

**Próximos Pasos**

1. Configurar variables de entorno
2. Mejorar manejo de errores
3. Añadir opciones de línea de comandos
4. Implementar logging más robusto