Documentación del Pipeline de Datos de Películas TMDB

Introducción

Este proyecto implementa un pipeline de datos completo que se encarga de extraer, procesar, almacenar y analizar información de películas proveniente de TMDB (The Movie Database). La solución utiliza un conjunto moderno de tecnologías de Big Data para procesar información en tiempo real, almacenarla de forma estructurada y generar análisis avanzados. El propósito principal es demostrar cómo diferentes componentes tecnológicos pueden integrarse para formar un flujo de datos coherente, desde la extracción inicial hasta el análisis avanzado con machine learning.

Arquitectura del Sistema

La arquitectura del sistema se basa en microservicios desplegados a través de Docker Compose, lo que facilita la escalabilidad, la modularidad y el mantenimiento del sistema. Cada componente se ejecuta en su propio contenedor, comunicándose entre sí a través de protocolos estandarizados.

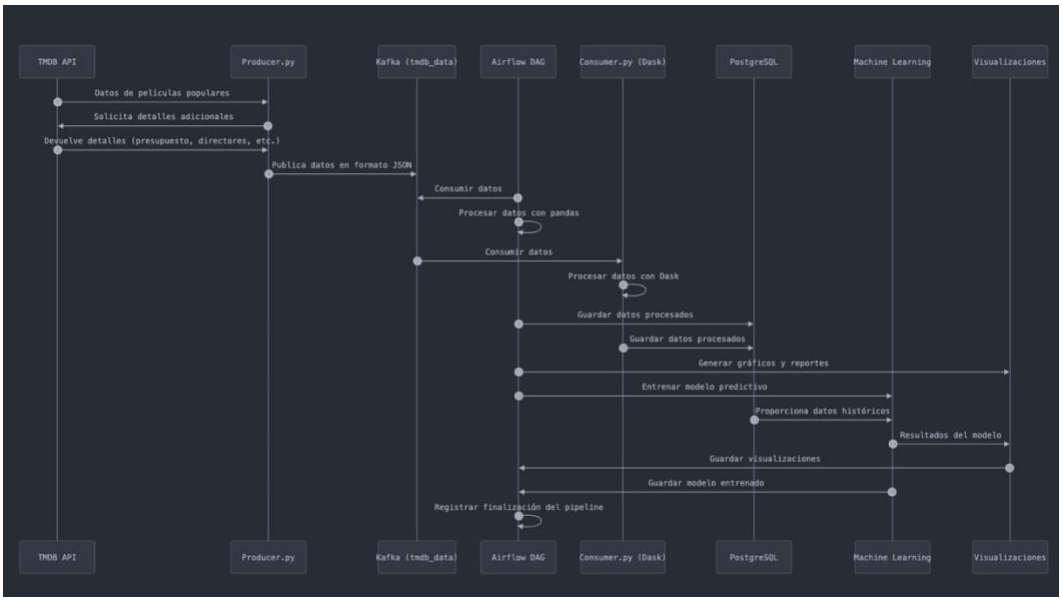
Los componentes principales del sistema son:

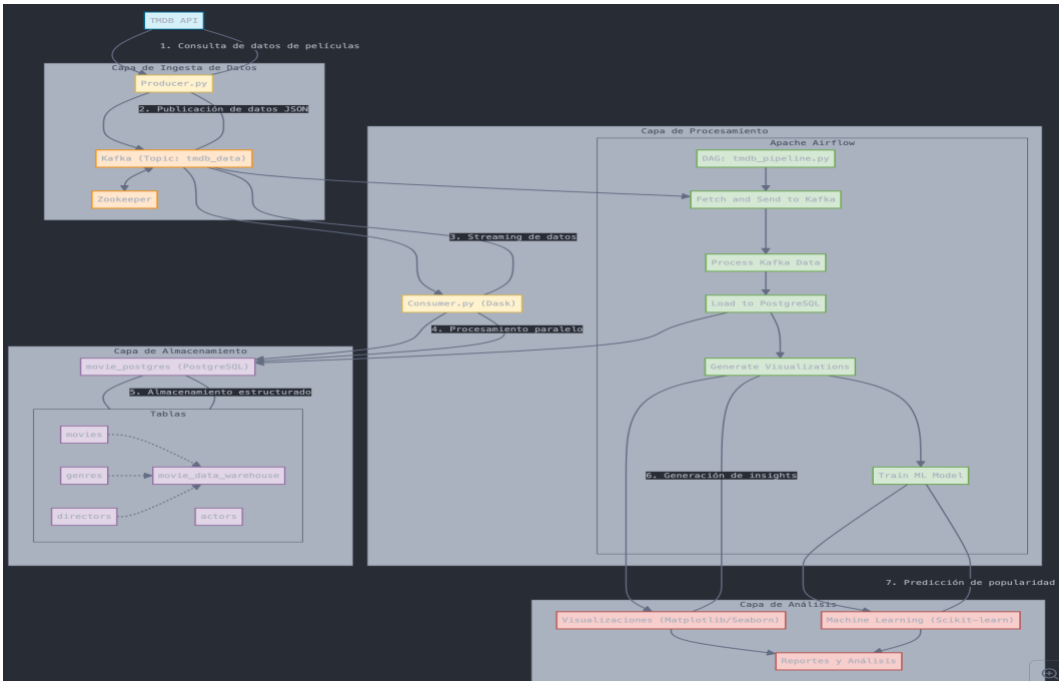
1. **Apache Kafka y Zookeeper**: Proporcionan una plataforma de streaming distribuida que permite la publicación y suscripción a flujos de datos. Kafka sirve como la columna vertebral para la transmisión de datos en tiempo real entre los diferentes componentes del sistema. Zookeeper se utiliza para la coordinación y gestión del clúster de Kafka.
2. **Apache Airflow**: Actúa como orquestador del pipeline, programando y ejecutando tareas en secuencia o en paralelo según las dependencias definidas. Airflow permite monitorizar el estado de cada tarea y reintentar automáticamente las tareas fallidas, proporcionando robustez al sistema.
3. **PostgreSQL**: Sirve como sistema de almacenamiento principal, utilizando un modelo relacional para datos estructurados y un almacén de datos (data warehouse) para facilitar el análisis. Se implementan dos instancias: una para Airflow y otra específica para los datos de películas.
4. **Dask**: Proporciona capacidades de computación paralela y distribuida para el procesamiento de datos, permitiendo manejar volúmenes significativos con mejor rendimiento que las soluciones secuenciales tradicionales.
5. **Python**: Es el lenguaje principal utilizado para implementar la lógica de negocio, incluyendo la interacción con APIs, el procesamiento de datos, y los modelos de machine learning.

Diagrama de Flujo de Datos

El flujo de datos sigue un proceso secuencial con etapas bien definidas:

1. **Extracción de Datos**: El proceso comienza cuando `producer.py` consulta la API de TMDB para obtener información detallada sobre películas populares. Esta información incluye detalles básicos (título, sinopsis, fecha de lanzamiento), así como información más específica como presupuesto, ingresos, géneros, directores y elenco principal.
2. **Publicación en Kafka**: Los datos extraídos se serializan en formato JSON y se publican en el topic "tmdb_data" de Kafka. Este enfoque de mensajería permite desacoplar la extracción de datos del procesamiento, proporcionando un buffer que absorbe picos de carga y permite que los datos se procesen a un ritmo adecuado.
3. **Procesamiento con Airflow**: Airflow orquesta el pipeline a través del DAG definido en `tmdb_pipeline.py`. Las tareas se ejecutan en secuencia, comenzando con la extracción de datos, continuando con el procesamiento, y finalizando con el análisis y visualización.
4. **Consumo y Procesamiento**: El componente `consumer.py` o la tarea correspondiente en Airflow consume los mensajes del topic de Kafka, deserializa los datos y realiza transformaciones utilizando Dask para paralelizar el procesamiento. Estas transformaciones incluyen limpieza de datos, cálculo de métricas derivadas y preparación para el almacenamiento.
5. **Almacenamiento en PostgreSQL**: Los datos procesados se almacenan en dos estructuras dentro de PostgreSQL:
    - Un modelo relacional normalizado con tablas para películas, géneros, directores, etc.
    - Una tabla de data warehouse para facilitar el análisis dimensional por género, director, año, etc.
6. **Análisis y Visualización**: Se generan análisis estadísticos y visualizaciones utilizando Matplotlib y Seaborn. Estos análisis incluyen distribuciones de géneros, correlaciones entre presupuesto e ingresos, y rankings de películas por popularidad.
7. **Modelo de Machine Learning**: Se entrena un modelo de regresión para predecir la popularidad de películas basándose en características como presupuesto, duración, y clasificación promedio. El modelo se evalúa utilizando métricas estándar y se guarda para su uso posterior.





Componentes del Sistema

docker-compose.yaml

El archivo `docker-compose.yaml` define la infraestructura completa del sistema. Especifica todos los servicios necesarios, sus dependencias, volúmenes,

configuraciones y conexiones de red. Este archivo es fundamental para el despliegue coherente del sistema en cualquier entorno compatible con Docker.

Los servicios configurados incluyen:

- **zookeeper**: Para la coordinación del clúster de Kafka
- **kafka**: Para la mensajería y streaming de datos
- **postgres**: Base de datos para Airflow
- **redis**: Para el almacenamiento en caché y como backend de resultados para Celery
- **movie_postgres**: Base de datos específica para almacenar datos de películas
- **airflow-webserver**, **airflow-scheduler**, **airflow-worker**: Componentes de Apache Airflow
- **airflow-init**: Servicio para inicializar Airflow

Cada servicio está configurado con parámetros específicos, como puertos, variables de entorno y volúmenes para persistencia de datos. El archivo también define una red Docker personalizada (`movie_pipeline`) para facilitar la comunicación entre contenedores.

setup_project.sh

El script `setup_project.sh` automatiza el proceso de configuración inicial del proyecto. Realiza las siguientes tareas:

- Verifica que Docker y Docker Compose estén instalados
- Crea la estructura de directorios necesaria
- Comprueba la existencia de archivos esenciales
- Detiene contenedores existentes que puedan interferir
- Inicia los servicios con Docker Compose
- Verifica que PostgreSQL y Kafka están funcionando correctamente
- Crea el topic de Kafka necesario
- Instala dependencias de Python en los contenedores de Airflow

Este script facilita enormemente el despliegue inicial del sistema, reduciendo la posibilidad de errores de configuración y garantizando que todos los componentes estén correctamente preparados.

install_deps.sh

El script `install_deps.sh` complementa al script de configuración, centrándose específicamente en la instalación de dependencias de Python en los contenedores de Airflow. Instala paquetes esenciales como Dask, Scikit-learn, Matplotlib, Pandas y Kafka-Python en los tres componentes principales de Airflow: webserver, scheduler y worker.

Este script es particularmente útil cuando se necesita actualizar o reinstalar dependencias después de que el sistema ya está en funcionamiento.

test_postgres.py

El script `test_postgres.py` verifica la conexión a la base de datos PostgreSQL específica para datos de películas. Realiza las siguientes acciones:

- Establece una conexión a PostgreSQL utilizando las credenciales configuradas
- Muestra información sobre la versión de PostgreSQL
- Lista las tablas existentes en el esquema público
- Crea una tabla de prueba para verificar los permisos de escritura

Este script es esencial para diagnosticar problemas de conexión a la base de datos antes de ejecutar el pipeline completo, lo que ayuda a identificar y resolver problemas de configuración tempranamente.

tmdb_pipeline.py

El archivo `tmdb_pipeline.py` define el Directed Acyclic Graph (DAG) de Airflow que orquesta todo el pipeline de datos. Este DAG está compuesto por las siguientes tareas principales:

1. **Iniciar Pipeline**: Crea directorios necesarios y prepara el entorno
2. **Obtener Datos y Enviar a Kafka**: Consulta la API de TMDB y publica los datos en Kafka
3. **Procesar Datos de Kafka**: Consume mensajes de Kafka y realiza transformaciones
4. **Cargar en PostgreSQL**: Almacena los datos procesados en la base de datos
5. **Generar Visualizaciones**: Crea gráficos y análisis visuales
6. **Entrenar Modelo ML**: Desarrolla un modelo predictivo usando los datos procesados
7. **Finalizar Pipeline**: Registra la finalización exitosa del proceso

Cada tarea está implementada como una función Python que utiliza las bibliotecas correspondientes para realizar su trabajo específico. Las tareas están conectadas en un flujo que define las dependencias entre ellas, garantizando que se ejecuten en el orden correcto.

producer.py

El script `producer.py` se encarga de extraer datos de la API de TMDB y enviarlos a Kafka. Sus funciones principales son:

1. **get_popular_movies**: Obtiene listas de películas populares paginadas desde la API de TMDB
2. **get_movie_details**: Recupera información detallada sobre una película específica, incluyendo créditos
3. **send_to_kafka**: Coordina la extracción de datos y su publicación en el topic "tmdb_data" de Kafka

El productor utiliza autenticación mediante token para acceder a la API de TMDB, y serializa los datos en formato JSON antes de enviarlos a Kafka. Implementa mecanismos de manejo de errores y limitación de velocidad para evitar sobrecargar la API.

consumer.py

El script `consumer.py` implementa un consumidor standalone para procesar datos desde Kafka utilizando Dask para paralelización. La clase principal `MovieAnalyzer` contiene los siguientes métodos:

1. **process_batch**: Procesa lotes de datos con Dask para aprovechar el paralelismo
2. **clean_dataframe**: Normaliza y limpia los datos, manejando valores nulos y formatos inconsistentes
3. **calculate_metrics**: Calcula métricas derivadas como ROI y categorías de popularidad
4. **save_to_database**: Almacena los datos procesados en PostgreSQL, gestionando las relaciones
5. **populate_data_warehouse**: Alimenta la tabla de data warehouse para análisis dimensional
6. **analyze_data**: Realiza consultas analíticas sobre los datos almacenados
7. **generate_visualizations**: Crea gráficos y visualizaciones basados en los análisis
8. **train_ml_model**: Entrena un modelo de machine learning para predecir la popularidad

Este componente implementa un modelo de datos relacional completo con clases SQLAlchemy para películas, géneros, directores, actores y compañías productoras, así como sus relaciones.

Implementación y Ejecución

Para ejecutar el pipeline completo, se deben seguir estos pasos en orden:

1. **Configuración del Entorno**:
   - Clonar el repositorio del proyecto
   - Ejecutar `setup_project.sh` para configurar el entorno inicial
2. **Verificación de Servicios**:

    - Comprobar que todos los contenedores Docker están ejecutándose correctamente
    - Verificar la conexión a PostgreSQL con `test_postgres.py`
    - Confirmar que el topic de Kafka ha sido creado correctamente
3. **Ejecución del Pipeline**:
    - Acceder a la UI de Airflow (http://localhost:8080) con usuario "admin" y contraseña "admin"
    - Activar el DAG "tmdb_pipeline"
    - Alternativamente, ejecutar el productor manualmente con `python data/producer.py`
    - Monitorear el progreso a través de la UI de Airflow o los logs
4. **Análisis de Resultados**:
    - Examinar las visualizaciones generadas en el directorio de salida
    - Revisar las métricas del modelo de machine learning
    - Consultar los datos almacenados en PostgreSQL para análisis adicionales
5. **Mantenimiento**:
    - Utilizar `install_deps.sh` para actualizar dependencias si es necesario
    - Monitorear los logs para detectar posibles problemas
    - Ajustar parámetros de configuración según sea necesario

Conclusiones y Beneficios

Este pipeline de datos ofrece varias ventajas significativas:

1. **Arquitectura Modular**: Cada componente puede ser desarrollado, actualizado y escalado independientemente, facilitando el mantenimiento y la evolución del sistema.
2. **Procesamiento en Tiempo Real**: La integración de Kafka permite procesar datos a medida que se generan, proporcionando resultados actualizados continuamente.
3. **Escalabilidad**: El uso de tecnologías como Docker, Kafka y Dask permite escalar horizontalmente para manejar volúmenes crecientes de datos.
4. **Automatización**: Airflow orquesta todo el proceso, ejecutando tareas programadas y manejando dependencias y fallos automáticamente.
5. **Análisis Avanzado**: La combinación de almacenamiento estructurado en PostgreSQL, análisis estadístico y machine learning proporciona insights valiosos sobre los datos de películas.

Este proyecto demuestra cómo las tecnologías modernas de big data pueden integrarse para formar un pipeline completo desde la extracción de datos hasta el análisis avanzado, sirviendo como base sólida para aplicaciones de análisis de datos en diversos dominios.