Monokera Prueba Técnica <> Ingeniero de Datos - Senior

Esta prueba técnica ha sido diseñada por **Monokera S.A.S.** y será utilizada únicamente con fines de evaluación para candidatos relacionados con procesos de admisión a la compañía. Prohibida su distribución y uso para otros fines ajenos a los establecidos por **Monokera S.A.S.**

</aside>

Contexto del Proyecto

Desarrollarás un pipeline de datos completo utilizando la API de Spaceflight News (https://api.spaceflightnewsapi.net/v4/docs/) para crear un sistema de análisis de tendencias en la industria espacial. El sistema procesará artículos de noticias, eventos y blogs para generar insights sobre la industria espacial.

Parte 1: Diseño de Arquitectura

Requisitos específicos:

- Ingesta diaria de nuevos artículos y eventos.
- Sistema de clasificación por temas.
- Almacenamiento histórico para análisis de tendencias.

Entregables Detallados:

1. **Diagrama de Arquitectura que incluya**:
    - Componentes AWS específicos con justificación.
    - Flujo de datos detallado.
    - Sistema de backup y recuperación.
2. **Documento Técnico que incluya**:
    - Estimación de volumen de datos.
    - Estrategia de almacenamiento y búsqueda.
    - Plan de contingencia.
    - Sistema de monitoreo.

Parte 2: Implementación del Pipeline

2.1 Extracción de Datos

Desarrolla un sistema de extracción que:

1. **Implemente conexión con Spaceflight News API:**

```
None
Endpoints requeridos:
- /articles (artículos con paginación)
- /blogs (blogs con paginación)
- /reports (reportes con paginación)
- /info (metadata de la API)
```

2.

```
Python
# Ejemplo de estructura de datos:{
    "id": int,
    "title": str,
    "url": str,
    "image_url": str,
    "news_site": str,
    "summary": str,
    "published_at": datetime,
    "updated_at": datetime,
    "featured": boolean,
    "launches": array,
    "events": array
}
```

3. **Funcionalidades Requeridas:**

    - Sistema de paginación eficiente.
    - Manejo de rate limits.
    - Deduplicación de artículos.
    - Sistema de logs detallado.
    - Tests unitarios.

2.2 Procesamiento con Spark

Implementa jobs de Spark que permitan en flujos posteriores calcular:

1. **Análisis de Contenido:**
    - Extracción de palabras clave y temas principales.
    - Identificación de entidades (compañías, personas, lugares).
    - Clasificación de artículos por tema.
2. **Análisis de Tendencias:**
    - Tendencias de temas por tiempo.
    - Análisis de fuentes de noticias más activas.
3. **Optimizaciones Requeridas:**
    - Particionamiento de datos históricos.
    - Caching de resultados frecuentes.

No es necesario aplicar modelos de Machine Learning, ni ser tediosos en la construcción de estos jobs. Esta parte de la implementación, deberá ayudarte para resolver la sección 3.

2.1 te ayudará en la capa de ingesta y 2.2 en la capa de procesamiento.

1. **Requerimientos:**
    - Manejo de fallos y retries
    - Validación de datos
    - Sistema de alertas
    - Documentación completa

Parte 3: Modelado y Análisis

3.1 Diseño del Data Warehouse

1. **Modelo Dimensional:**

```sql
-- Ejemplo de esquema sugerido:
CREATE TABLE dim_news_source (
    source_id INT PRIMARY KEY,
    name VARCHAR(255),
    url VARCHAR(255),
    reliability_score FLOAT);
CREATE TABLE dim_topic (
    topic_id INT PRIMARY KEY,
    name VARCHAR(255),
    category VARCHAR(255)
);
CREATE TABLE fact_article (
    article_id INT PRIMARY KEY,
    source_id INT REFERENCES dim_news_source(source_id),
    topic_id INT REFERENCES dim_topic(topic_id),
```

```
    published_at TIMESTAMP,
    -- otros campos relevantes);
```

1. **Optimizaciones:**
    - Índices apropiados
    - Particionamiento temporal
    - Estrategia de actualización
2. Las tables anteriores, son un ejemplo y no necesariamente deben ser así. Ajusta según análisis y necesidades que has observado.

3.2 Análisis SQL

Desarrollar queries para:

1. **Análisis de Tendencias:**
    1. Tendencias de temas por mes
    2. Fuentes más influyentes

3.1 te ayudará en la capa de almacenamiento y servicio y 3.2 en la capa de consumo.

Parte 4: Pipeline con Airflow

Crear DAG que incluya algo como:

1. **Tasks Específicas:**

Python
```python
def spacenews_pipeline():
    # Extracción
    extract_articles = PythonOperator(...)
    extract_blogs = PythonOperator(...)
    extract_reports = PythonOperator(...)
    # Procesamiento
    clean_and_deduplicate = SparkSubmitOperator(...)
    perform_analysis = SparkSubmitOperator(...)
    identify_topics = SparkSubmitOperator(...)
    # Carga y Análisis
    load_processed_data = PythonOperator(...)
    generate_daily_insights = PythonOperator(...)
```

```
    update_dashboards = PythonOperator(...)
    # Dependencias
    [extract_articles, extract_blogs, extract_reports] >>
clean_and_deduplicate
    clean_and_deduplicate >> [perform_analysis,
identify_topics]
    [perform_sentiment_analysis, identify_topics] >>
load_processed_data
    load_processed_data >> generate_daily_insights >>
update_dashboards
```

Tener en cuenta que no es necesario seguir los mismos nombres y operadores de las task descritas anteriormente.

Entrega:

1. Repositorio Git con:
    - Código documentado
    - Tests unitarios
    - README detallado
    - Documentación de arquitectura
2. Scripts SQL
3. Documento técnico
4. Presentación

Consideraciones extras:

- Puedes usar herramientas de IA para desarrollar tu prueba.
- Si es posible, detalla los tiempos que te tomo resolver cada una de las secciones descritas en la prueba. Esta información no afecta en las calificaciones finales de la prueba.
- El sistema de clasificación será:
    - **Arquitectura (20 pts)**
    - **Implementación (40 pts)**
    - **Modelado y Análisis (30 pts)**
    - **Sustentación (10 pts)**
- Para la sustentación de la prueba tendrás 30 minutos, en donde se contemplarán la realización de preguntas por los integrantes del equipo. Prepara previamente lo que se va a presentar teniendo en cuenta:
    - Deberás hacer una demostración del proyecto
    - Explicar decisiones técnicas bajo tu experiencia.
    - Compartiras un análisis de limitaciones y mejoras del proyecto.
    - Tendremos de manera dinámica preguntas y respuestas

- Si en lugar de Airflow y Spark, deseas resolver el problema con otras tecnologías, no hay problema, pero deberá argumentarse técnicamente las ventajas del cambio.
- Si considera que dentro del proyecto hace falta contemplar algo a nivel técnico que mejore el proyecto, tiene la libertad de implementarlo. Si desea excluir algo porque lo considera irrelevante, también puede hacerlo.