Aquí tienes una guía **paso a paso** para implementar un **sistema de procesamiento en tiempo real usando Kafka y Dask** con datos de **Alpha Vantage API**. Adaptado para **dummies** con explicaciones claras y código.

**🔹 Paso 1: Preparar el entorno**

**📌 Instalación de dependencias**

Asegúrate de tener instalado lo siguiente en tu entorno:

pip install kafka-python dask pandas matplotlib alpha\_vantage

Si usas **Docker**, también instala y configura **Kafka**:

docker-compose up -d

⚠️ **Verifica que Kafka esté corriendo:**

docker ps

Debería aparecer un contenedor con **Apache Kafka**.

**🔹 Paso 2: Obtener datos en tiempo real de la API**

**📌 Regístrate en Alpha Vantage**

1. Ve a [Alpha Vantage](https://www.alphavantage.co/)
2. Regístrate y obtén tu **API Key**
3. Guarda esta API Key, la usaremos en el código

**🔹 Paso 3: Implementar Kafka Producer**

El **Producer** obtendrá datos de Alpha Vantage y los enviará a Kafka.

📄 **Crea el archivo** producer.py en: 📂 C:\Users\JONANNA\Documents\MINE\_III\Big\_Data\_3

from kafka import KafkaProducer

import json

import time

from alpha\_vantage.timeseries import TimeSeries

# Configuración de Alpha Vantage

API\_KEY = "TU\_API\_KEY" # Sustituye con tu clave real

SYMBOL = "AAPL" # Acción de Apple (puedes cambiarlo)

INTERVAL = "1min"

# Configuración de Kafka

TOPIC = "stock\_data"

KAFKA\_SERVER = "localhost:9092"

# Inicializar el productor de Kafka

producer = KafkaProducer(

bootstrap\_servers=KAFKA\_SERVER,

value\_serializer=lambda v: json.dumps(v).encode("utf-8"),

)

# Función para obtener datos

def fetch\_stock\_data():

ts = TimeSeries(key=API\_KEY, output\_format="json")

data, meta = ts.get\_intraday(symbol=SYMBOL, interval=INTERVAL, outputsize="compact")

latest\_time = list(data.keys())[0]

stock\_info = {"time": latest\_time, "price": data[latest\_time]["1. open"]}

return stock\_info

# Enviar datos a Kafka cada 10 segundos

while True:

stock\_data = fetch\_stock\_data()

producer.send(TOPIC, stock\_data)

print(f"📤 Enviado: {stock\_data}")

time.sleep(10) # Espera 10 segundos para la próxima consulta

**✅ Explicación:**

* Se conecta a Alpha Vantage y obtiene el último precio de Apple.
* Lo envía a un **Kafka Topic** (stock\_data).
* Se ejecuta cada 10 segundos.

**🔹 Paso 4: Implementar Kafka Consumer + Procesamiento con Dask**

📄 **Crea el archivo** consumer.py

from kafka import KafkaConsumer

import json

import pandas as pd

import dask.dataframe as dd

# Configuración de Kafka

TOPIC = "stock\_data"

KAFKA\_SERVER = "localhost:9092"

# Crear consumidor

consumer = KafkaConsumer(

TOPIC,

bootstrap\_servers=KAFKA\_SERVER,

value\_deserializer=lambda v: json.loads(v.decode("utf-8")),

)

# Lista para almacenar datos antes de procesar

data\_list = []

print("🟢 Esperando datos...")

for message in consumer:

stock\_data = message.value

print(f"📥 Recibido: {stock\_data}")

data\_list.append(stock\_data)

# Si hay más de 10 registros, procesamos con Dask

if len(data\_list) >= 10:

df = pd.DataFrame(data\_list) # Convertir a DataFrame

ddf = dd.from\_pandas(df, npartitions=2) # Convertir a Dask DataFrame

# Cálculo: promedio del precio

avg\_price = ddf["price"].mean().compute()

print(f"📊 Promedio del precio: {avg\_price}")

data\_list = [] # Limpiar lista para nueva tanda

**✅ Explicación:**

* Recibe los datos de Kafka.
* Los almacena temporalmente.
* Usa **Dask** para calcular el **promedio** cada 10 registros.

**🔹 Paso 5: Almacenamiento y Visualización**

📄 **Crea el archivo** visualization.py

import pandas as pd

import matplotlib.pyplot as plt

# Cargar datos procesados

df = pd.read\_csv("processed\_stock\_data.csv")

# Graficar precios de la acción

plt.figure(figsize=(10,5))

plt.plot(df["time"], df["price"], marker="o", linestyle="-", color="b")

plt.xlabel("Tiempo")

plt.ylabel("Precio")

plt.title("Evolución del Precio de la Acción")

plt.xticks(rotation=45)

plt.show()

**✅ Explicación:**

* Carga datos almacenados en CSV.
* Genera una gráfica de **evolución del precio**.

**🔹 Paso 6: Automatizar ejecución**

Ejecuta los programas en orden:

1️⃣ **Inicia Kafka** (Si no está corriendo):

docker-compose up -d

2️⃣ **Ejecuta el producer (envía datos)**

python producer.py

3️⃣ **Ejecuta el consumer (procesa datos)**

python consumer.py

4️⃣ **Ejecuta la visualización**

python visualization.py

**🔹 Productos que debes entregar**

✅ **Código fuente** de los tres scripts:

* producer.py
* consumer.py
* visualization.py

✅ **Explicación escrita** con los pasos de instalación y ejecución.

✅ **Gráfica generada** de los datos procesados.

✅ **Reporte final** con análisis de los datos.

**🚀 ¡Listo! Has implementado un sistema de procesamiento en tiempo real con Kafka y Dask 🎯**