Veo que la tarea consiste en implementar un sistema de procesamiento en tiempo real usando **Kafka** y **Dask** con datos de una API pública. Además, mencionaste usar la base de **Poetry de contratos del SECOP** y tu usuario de GitHub es [djsoler](https://github.com/djsoler).

Voy a explicarlo paso a paso desde cero, estilo *dummies*:

**✅ 1. Configuración del Entorno**

**🐍 Instalar Dependencias con Poetry**

1. Abre una terminal y navega a la carpeta donde trabajarás:
2. cd C:\Users\JONANNA\Documents\MINE\_III\big\_data
3. Inicializa el proyecto con Poetry:
4. poetry init
   * Responde las preguntas o presiona Enter para los valores por defecto.
5. Edita el archivo pyproject.toml y agrega las siguientes dependencias:
6. [tool.poetry.dependencies]
7. python = "^3.10"
8. kafka-python = "^2.0.2"
9. dask = "^2024.1.1"
10. pandas = "^2.1.4"
11. matplotlib = "^3.8.0"
12. requests = "^2.31.0"
13. Instala las dependencias:
14. poetry install

**🐋 2. Configurar Kafka con Docker**

Ya que tienes Docker Desktop instalado, vamos a iniciar Kafka usando Docker Compose.

1. Crea un archivo llamado docker-compose.yml en la carpeta del proyecto y agrega lo siguiente:

version: '3.8'

services:

zookeeper:

image: zookeeper:3.7.1

container\_name: zookeeper

ports:

- "2181:2181"

environment:

ZOO\_MY\_ID: 1

ZOO\_PORT: 2181

ZOO\_SERVERS: server.1=zookeeper:2888:3888

kafka:

image: apache/kafka:3.7.1

container\_name: kafka

ports:

- "9092:9092"

environment:

KAFKA\_BROKER\_ID: 1

KAFKA\_ZOOKEEPER\_CONNECT: zookeeper:2181

KAFKA\_ADVERTISED\_LISTENERS: PLAINTEXT://localhost:9092

KAFKA\_OFFSETS\_TOPIC\_REPLICATION\_FACTOR: 1

depends\_on:

- zookeeper

1. Inicia los contenedores:

docker compose up -d

Esto arrancará Zookeeper y Kafka en segundo plano.

**🌐 3. Consumir Datos de una API con Kafka Producer**

Vamos a crear un productor que lea datos de una API y los envíe a un **Topic** de Kafka.

1. Crea un archivo producer.py:

from kafka import KafkaProducer

import requests

import json

import time

API\_URL = "https://api.open-meteo.com/v1/forecast?latitude=4.6097&longitude=-74.0817&hourly=temperature\_2m"

KAFKA\_TOPIC = "weather\_data"

producer = KafkaProducer(

bootstrap\_servers="localhost:9092",

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

)

while True:

response = requests.get(API\_URL)

if response.status\_code == 200:

data = response.json()

producer.send(KAFKA\_TOPIC, data)

print("Datos enviados a Kafka:", data)

time.sleep(10) # Espera 10 segundos antes de la siguiente solicitud

Este código envía los datos meteorológicos de Bogotá al topic weather\_data cada 10 segundos.

**💾 4. Consumir y Procesar Datos con Kafka Consumer y Dask**

Ahora crearemos un consumidor que reciba los datos y los procese con Dask.

1. Crea un archivo consumer.py:

from kafka import KafkaConsumer

import json

import pandas as pd

import dask.dataframe as dd

import datetime

KAFKA\_TOPIC = "weather\_data"

consumer = KafkaConsumer(

KAFKA\_TOPIC,

bootstrap\_servers="localhost:9092",

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

)

data\_list = []

for message in consumer:

data = message.value

temp\_data = data["hourly"]["temperature\_2m"]

df = pd.DataFrame(temp\_data, columns=["temperature"])

df["timestamp"] = pd.to\_datetime(datetime.datetime.now())

data\_list.append(df)

# Procesamiento con Dask

if len(data\_list) >= 5: # Procesa cada 5 mensajes

ddf = dd.from\_pandas(pd.concat(data\_list), npartitions=2)

result = ddf.groupby("timestamp").mean().compute()

result.to\_parquet("processed\_weather.parquet")

print("Datos procesados y guardados")

data\_list = []

Este consumidor procesa cada 5 mensajes y guarda los datos procesados en formato **Parquet**.

**📊 5. Visualizar los Datos con Pandas y Matplotlib**

Crea un archivo visualization.py para mostrar los datos guardados.

import pandas as pd

import matplotlib.pyplot as plt

df = pd.read\_parquet("processed\_weather.parquet")

df.plot(x="timestamp", y="temperature", kind="line", title="Temperatura en Bogotá")

plt.show()

**🚀 6. Ejecutar Todo el Proyecto**

1. Inicia Docker con Kafka:

docker compose up -d

1. Abre dos terminales:

* **Terminal 1:** Ejecuta el productor

poetry run python producer.py

* **Terminal 2:** Ejecuta el consumidor

poetry run python consumer.py

1. Después de unos minutos, ejecuta la visualización:

poetry run python visualization.py

**🗂️ 7. Subir el Proyecto a GitHub**

1. Crea un repositorio en [tu cuenta de GitHub](https://github.com/djsoler).
2. En la terminal, configura el repositorio:

git init

git add .

git commit -m "Proyecto Kafka y Dask"

git branch -M main

git remote add origin https://github.com/djsoler/nombre-del-repositorio.git

git push -u origin main

Si quieres que adapte esto a la **base de Poetry de contratos del SECOP**, avísame y lo integro en los scripts. 😊

Veo que la tarea consiste en implementar un sistema de procesamiento en tiempo real usando **Kafka** y **Dask** con datos de una API pública. Además, mencionaste usar la base de **Poetry de contratos del SECOP** y tu usuario de GitHub es [djsoler](https://github.com/djsoler).

Voy a explicarlo paso a paso desde cero, estilo *dummies*:

**✅ 1. Configuración del Entorno**

**🐍 Instalar Dependencias con Poetry**

1. Abre una terminal y navega a la carpeta donde trabajarás:
2. cd C:\Users\JONANNA\Documents\MINE\_III\big\_data
3. Inicializa el proyecto con Poetry:
4. poetry init
   * Responde las preguntas o presiona Enter para los valores por defecto.
5. Edita el archivo pyproject.toml y agrega las siguientes dependencias:
6. [tool.poetry.dependencies]
7. python = "^3.10"
8. kafka-python = "^2.0.2"
9. dask = "^2024.1.1"
10. pandas = "^2.1.4"
11. matplotlib = "^3.8.0"
12. requests = "^2.31.0"
13. Instala las dependencias:
14. poetry install

**🐋 2. Configurar Kafka con Docker**

Ya que tienes Docker Desktop instalado, vamos a iniciar Kafka usando Docker Compose.

1. Crea un archivo llamado docker-compose.yml en la carpeta del proyecto y agrega lo siguiente:

version: '3.8'

services:

zookeeper:

image: zookeeper:3.7.1

container\_name: zookeeper

ports:

- "2181:2181"

environment:

ZOO\_MY\_ID: 1

ZOO\_PORT: 2181

ZOO\_SERVERS: server.1=zookeeper:2888:3888

kafka:

image: apache/kafka:3.7.1

container\_name: kafka

ports:

- "9092:9092"

environment:

KAFKA\_BROKER\_ID: 1

KAFKA\_ZOOKEEPER\_CONNECT: zookeeper:2181

KAFKA\_ADVERTISED\_LISTENERS: PLAINTEXT://localhost:9092

KAFKA\_OFFSETS\_TOPIC\_REPLICATION\_FACTOR: 1

depends\_on:

- zookeeper

1. Inicia los contenedores:

docker compose up -d

Esto arrancará Zookeeper y Kafka en segundo plano.

**🌐 3. Consumir Datos de una API con Kafka Producer**

Vamos a crear un productor que lea datos de una API y los envíe a un **Topic** de Kafka.

1. Crea un archivo producer.py:

from kafka import KafkaProducer

import requests

import json

import time

API\_URL = "https://api.open-meteo.com/v1/forecast?latitude=4.6097&longitude=-74.0817&hourly=temperature\_2m"

KAFKA\_TOPIC = "weather\_data"

producer = KafkaProducer(

bootstrap\_servers="localhost:9092",

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

)

while True:

response = requests.get(API\_URL)

if response.status\_code == 200:

data = response.json()

producer.send(KAFKA\_TOPIC, data)

print("Datos enviados a Kafka:", data)

time.sleep(10) # Espera 10 segundos antes de la siguiente solicitud

Este código envía los datos meteorológicos de Bogotá al topic weather\_data cada 10 segundos.

**💾 4. Consumir y Procesar Datos con Kafka Consumer y Dask**

Ahora crearemos un consumidor que reciba los datos y los procese con Dask.

1. Crea un archivo consumer.py:

from kafka import KafkaConsumer

import json

import pandas as pd

import dask.dataframe as dd

import datetime

KAFKA\_TOPIC = "weather\_data"

consumer = KafkaConsumer(

KAFKA\_TOPIC,

bootstrap\_servers="localhost:9092",

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

)

data\_list = []

for message in consumer:

data = message.value

temp\_data = data["hourly"]["temperature\_2m"]

df = pd.DataFrame(temp\_data, columns=["temperature"])

df["timestamp"] = pd.to\_datetime(datetime.datetime.now())

data\_list.append(df)

# Procesamiento con Dask

if len(data\_list) >= 5: # Procesa cada 5 mensajes

ddf = dd.from\_pandas(pd.concat(data\_list), npartitions=2)

result = ddf.groupby("timestamp").mean().compute()

result.to\_parquet("processed\_weather.parquet")

print("Datos procesados y guardados")

data\_list = []

Este consumidor procesa cada 5 mensajes y guarda los datos procesados en formato **Parquet**.

**📊 5. Visualizar los Datos con Pandas y Matplotlib**

Crea un archivo visualization.py para mostrar los datos guardados.

import pandas as pd

import matplotlib.pyplot as plt

df = pd.read\_parquet("processed\_weather.parquet")

df.plot(x="timestamp", y="temperature", kind="line", title="Temperatura en Bogotá")

plt.show()

**🚀 6. Ejecutar Todo el Proyecto**

1. Inicia Docker con Kafka:

docker compose up -d

1. Abre dos terminales:

* **Terminal 1:** Ejecuta el productor

poetry run python producer.py

* **Terminal 2:** Ejecuta el consumidor

poetry run python consumer.py

1. Después de unos minutos, ejecuta la visualización:

poetry run python visualization.py

**🗂️ 7. Subir el Proyecto a GitHub**

1. Crea un repositorio en [tu cuenta de GitHub](https://github.com/djsoler).
2. En la terminal, configura el repositorio:

git init

git add .

git commit -m "Proyecto Kafka y Dask"

git branch -M main

git remote add origin https://github.com/djsoler/nombre-del-repositorio.git

git push -u origin main

Si quieres que adapte esto a la **base de Poetry de contratos del SECOP**, avísame y lo integro en los scripts. 😊

CON POETRY

# Proyecto: Procesamiento de Datos del SECOP en Tiempo Real con Kafka y Dask

## Estructura del Proyecto

```

C:\Users\JONANNA\Documents\MINE\_III\big\_data

├── docker-compose.yml

├── pyproject.toml

├── producer.py

├── consumer\_dask.py

├── storage.py

└── dashboard.py

```

---

## 1. docker-compose.yml

```yaml

version: '2'

services:

zookeeper:

image: zookeeper:latest

ports:

- "2181:2181"

kafka:

image: apache/kafka:3.7.1

ports:

- "9092:9092"

environment:

KAFKA\_BROKER\_ID: 1

KAFKA\_ZOOKEEPER\_CONNECT: zookeeper:2181

KAFKA\_ADVERTISED\_LISTENERS: PLAINTEXT://localhost:9092

depends\_on:

- zookeeper

```

---

## 2. pyproject.toml

```toml

[tool.poetry]

name = "secop\_real\_time"

version = "0.1.0"

description = "Procesamiento de datos de contratos del SECOP en tiempo real usando Kafka y Dask"

authors = ["Johanna Soler"]

[tool.poetry.dependencies]

python = "^3.10"

kafka-python = "^2.0.2"

dask = "^2024.3.1"

pandas = "^2.2.0"

pyarrow = "^15.0.0"

matplotlib = "^3.8.0"

dash = "^2.15.0"

[build-system]

requires = ["poetry-core"]

build-backend = "poetry.core.masonry.api"

```

---

## 3. producer.py

```python

from kafka import KafkaProducer

import pandas as pd

import json

import time

producer = KafkaProducer(

bootstrap\_servers='localhost:9092',

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

)

df = pd.read\_csv('contratos\_secop.csv')

df = df.to\_dict(orient='records')

for row in df:

producer.send('secop\_contratos', row)

print(f"Enviado: {row}")

time.sleep(1)

producer.close()

```

---

## 4. consumer\_dask.py

```python

from kafka import KafkaConsumer

import dask.dataframe as dd

import pandas as pd

import json

consumer = KafkaConsumer(

'secop\_contratos',

bootstrap\_servers='localhost:9092',

auto\_offset\_reset='earliest',

group\_id='grupo\_contratos',

value\_deserializer=lambda x: json.loads(x.decode('utf-8'))

)

datos = []

for mensaje in consumer:

datos.append(mensaje.value)

if len(datos) >= 10:

ddf = dd.from\_pandas(pd.DataFrame(datos), npartitions=2)

resultado = ddf.groupby('Departamento')['Valor\_Contrato'].sum().compute()

print(resultado)

datos = []

```

---

## 5. storage.py

```python

import pandas as pd

def guardar\_parquet(datos, archivo='resultados\_contratos.parquet'):

df = pd.DataFrame(datos)

df.to\_parquet(archivo, engine='pyarrow')

print(f"Datos guardados en {archivo}")

```

---

## 6. dashboard.py

```python

from dash import Dash, html, dcc

import pandas as pd

import plotly.express as px

app = Dash(\_\_name\_\_)

df = pd.read\_parquet('resultados\_contratos.parquet')

fig = px.bar(df, x='Departamento', y='Valor\_Contrato', title='Valor Total de Contratos por Departamento')

app.layout = html.Div(children=[

html.H1(children='Reporte de Contratos SECOP'),

dcc.Graph(figure=fig)

])

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

```

---

### Instrucciones para Ejecutar

1. \*\*Iniciar Docker:\*\*

```bash

cd C:\Users\JONANNA\Documents\MINE\_III\big\_data

docker-compose up -d

```

2. \*\*Instalar dependencias con Poetry:\*\*

```bash

poetry install

```

3. \*\*Ejecutar Kafka Producer y Consumer:\*\*

```bash

poetry run python producer.py

poetry run python consumer\_dask.py

```

4. \*\*Mostrar el Dashboard:\*\*

```bash

poetry run python dashboard.py

```

---

¿Te gustaría agregar un archivo README.md con las instrucciones de uso? 😊