**Lab Exercise 18- Visualization Data Pipeline in Metaflow**

Lab exercise to create a data pipeline using Metaflow that generates a chart using **Matplotlib**. This pipeline will simulate data generation, process the data, and generate a chart at the end using Matplotlib.

**Objective:**

* Build a data pipeline using Metaflow.
* Generate a chart at the end of the pipeline using Matplotlib.

**Steps:**

1. Define a flow that simulates data generation, processing, and visualization.
2. Use Matplotlib to generate and save a chart.
3. Add steps to save the chart as an image file.

**Step-by-Step Lab Exercise**

**1. Install Required Libraries**

First, ensure you have the required packages installed:

pip install matplotlib metaflow

**2. Create the Data Pipeline with Matplotlib Chart Generation**

**File: data\_pipeline\_with\_chart.py**

# data\_pipeline\_with\_chart.py

import matplotlib.pyplot as plt

import numpy as np

from metaflow import FlowSpec, step, current

class DataPipelineFlow(FlowSpec):

@step

def start(self):

"""

Step 1: Simulate Data Generation.

We simulate some random data for demonstration purposes.

"""

print("Generating random data...")

np.random.seed(42)

self.data = np.random.normal(0, 1, 1000) # 1000 random data points

print(f"Data generated: {self.data[:5]}...") # Print first 5 data points

self.next(self.process\_data)

@step

def process\_data(self):

"""

Step 2: Process the data.

For this example, let's calculate the histogram of the data.

"""

print("Processing data (creating histogram)...")

self.hist, self.bin\_edges = np.histogram(self.data, bins=30)

print(f"Histogram calculated: {self.hist[:5]}...") # Print first 5 histogram bins

self.next(self.generate\_chart)

@step

def generate\_chart(self):

"""

Step 3: Generate a chart using Matplotlib.

We will create a histogram and save it as an image.

"""

print("Generating chart...")

# Plot the histogram

plt.figure(figsize=(8, 6))

plt.hist(self.data, bins=30, color='skyblue', edgecolor='black')

plt.title('Histogram of Generated Data')

plt.xlabel('Data Values')

plt.ylabel('Frequency')

# Save the chart as a PNG image

chart\_filename = f'{current.flow\_name}\_{current.run\_id}\_histogram.png'

plt.savefig(chart\_filename)

print(f"Chart saved as {chart\_filename}")

self.next(self.end)

@step

def end(self):

"""

Final step: Notify that the flow has completed.

"""

print("Flow finished successfully!")

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

DataPipelineFlow()

**3. How the Code Works**

* **Step 1 (start)**: Simulates the generation of 1000 random data points from a normal distribution.
* **Step 2 (process\_data)**: Processes the generated data by calculating its histogram using numpy.histogram().
* **Step 3 (generate\_chart)**: Uses **Matplotlib** to generate a histogram of the data and saves the chart as a .png file.
* **Step 4 (end)**: The flow finishes and prints a success message.

**4. Running the Data Pipeline**

To run the pipeline and generate the chart, use the following command:

python data\_pipeline\_with\_chart.py run