**Lab Exercise-1 Using Metaflow with MinIO**

**Objective**: Create a simple Metaflow flow that uploads a CSV file to a MinIO bucket and retrieves it.

**Prerequisites**

1. **Docker**: Ensure Docker Desktop is installed and running.
2. **MinIO**: Run a MinIO server locally. You can do this with Docker:

docker run -p 9000:9000 -p 9001:9001 --name minio \

-e "MINIO\_ACCESS\_KEY=minioadmin" \

-e "MINIO\_SECRET\_KEY=minioadmin" \

minio/minio server /data --console-address ":9001"

Here's how you can organize the code for the Metaflow exercise using a MinIO bucket with a proper directory structure. This structure will help keep your project organized.

**Directory Structure**

metaflow\_minio\_project/

│

├── data/

│ └── sample\_data.csv

│

├── output/

│ └── downloaded\_data.csv

│

├── minio\_metaflow.py

└── requirements.txt

**Step 1: Create Directory Structure**

You can create this structure using the following commands in your terminal:

mkdir -p metaflow\_minio\_project/data

mkdir -p metaflow\_minio\_project/output

cd metaflow\_minio\_project

touch minio\_metaflow.py requirements.txt

**Step 2: Add Dependencies**

Open requirements.txt and add the following lines:

metaflow

pandas

minio

**Step 3: Main Metaflow Script**

Now, open ***minio\_metaflow.py*** and add the following code:

from metaflow import FlowSpec, step

import pandas as pd

from minio import Minio

from minio.error import S3Error

import os

class MinIOFlow(FlowSpec):

@step

def start(self):

"""Generate sample data and upload to MinIO."""

# Create a simple DataFrame

data = {'Name': ['Alice', 'Bob', 'Charlie'],

'Age': [24, 30, 22]}

self.df = pd.DataFrame(data)

# Save to CSV in the data directory

self.data\_path = 'data/sample\_data.csv'

self.df.to\_csv(self.data\_path, index=False)

self.next(self.upload\_to\_minio)

@step

def upload\_to\_minio(self):

"""Upload the CSV file to MinIO."""

# Instantiate MinIO client directly in this step

minio\_client = Minio('localhost:9000',

access\_key='minioadmin',

secret\_key='minioadmin',

secure=False)

# Create bucket if it doesn't exist

try:

if not minio\_client.bucket\_exists('mybucket'):

minio\_client.make\_bucket('mybucket')

except S3Error as e:

print("Error occurred: ", e)

# Upload the file

try:

minio\_client.fput\_object('mybucket', 'sample\_data.csv', self.data\_path)

print("File uploaded successfully.")

except S3Error as e:

print("Error occurred: ", e)

self.next(self.retrieve\_from\_minio)

@step

def retrieve\_from\_minio(self):

"""Retrieve the CSV file from MinIO."""

# Instantiate MinIO client again in this step

minio\_client = Minio('localhost:9000',

access\_key='minioadmin',

secret\_key='minioadmin',

secure=False)

# Download the file to output directory

self.output\_path = 'output/downloaded\_data.csv'

minio\_client.fget\_object('mybucket', 'sample\_data.csv', self.output\_path)

print("File downloaded successfully.")

# Read the downloaded file

self.downloaded\_df = pd.read\_csv(self.output\_path)

print("Downloaded DataFrame:")

print(self.downloaded\_df)

self.next(self.end)

@step

def end(self):

"""End of the flow."""

print("Flow finished!")

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

MinIOFlow()

**Step 4: Running the Flow**

1. **Install Dependencies**: First, ensure you have all the required packages installed. You can do this by running:

pip install -r requirements.txt

1. **Set Up MinIO**: Ensure your MinIO server is running. You can start it with the Docker command provided earlier.
2. **Run the Flow**: In your terminal, navigate to the metaflow\_minio\_project directory and run the flow:

python minio\_metaflow.py run

**Step 5: Check Results**

After running the flow, check the following:

* The data/sample\_data.csv file will be created with the sample data.
* The output/downloaded\_data.csv file will contain the same data downloaded from the MinIO bucket.
* You can also access the MinIO console at http://localhost:9001 to verify the file upload.

This directory structure helps keep your project organized, separating data inputs and outputs from your code.