pip install ibm-cos-sdk

import ibm\_boto3

from ibm\_botocore.client import Config, ClientError

# Configuration

cos = ibm\_boto3.client(

's3',

ibm\_api\_key\_id='your-api-key', # Replace with your API key

ibm\_service\_instance\_id='your-service-instance-id', # Replace with your service instance ID

config=Config(signature\_version='oauth'),

endpoint\_url='https://s3.<region>.cloud-object-storage.appdomain.cloud' # Replace with your region

)

def backup\_file(file\_path, bucket\_name, object\_name):

"""

Backs up a file to IBM Cloud Object Storage.

:param file\_path: Path to the file to back up.

:param bucket\_name: Name of the bucket.

:param object\_name: Name of the object to create in the bucket.

"""

try:

# Upload the file

cos.upload\_file(Filename=file\_path, Bucket=bucket\_name, Key=object\_name)

print(f"File '{file\_path}' successfully uploaded to bucket '{bucket\_name}' as '{object\_name}'.")

except ClientError as e:

print(f"Error uploading file: {e}")

# Usage

file\_path = "local\_data.csv" # Local file path

bucket\_name = "my-storage-bucket" # Replace with your bucket name

object\_name = "backup\_data/local\_data.csv" # Object name in the bucket

# Backup file to IBM Cloud

backup\_file(file\_path, bucket\_name, object\_name)

from flask import Flask, jsonify

import ibm\_boto3

from ibm\_botocore.client import Config, ClientError

app = Flask(\_\_name\_\_)

# Initialize the COS client with proper configuration

cos = ibm\_boto3.client(

's3',

ibm\_api\_key\_id='your-api-key', # Replace with your IBM API key

ibm\_service\_instance\_id='your-service-instance-id', # Replace with your service instance ID

config=Config(signature\_version='oauth'),

endpoint\_url='https://s3.<region>.cloud-object-storage.appdomain.cloud' # Replace with your region endpoint

)

@app.route('/restore/<bucket\_name>/<file\_name>', methods=['GET'])

def restore\_file(bucket\_name, file\_name):

"""

Endpoint to restore (download) a file from IBM Cloud Object Storage.

:param bucket\_name: Name of the bucket.

:param file\_name: Name of the file to restore.

:return: JSON response indicating success or failure.

"""

try:

# Attempt to download the file from the bucket

cos.download\_file(bucket\_name, file\_name, f'restored\_{file\_name}')

return jsonify({"message": f"File '{file\_name}' restored successfully."}), 200

except ClientError as e:

# Handle any errors from the IBM COS client

return jsonify({"error": f"Error restoring file: {e}"}), 500

except Exception as e:

# Handle other generic exceptions

return jsonify({"error": f"An unexpected error occurred: {e}"}), 500

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

# Run the Flask app in debug mode

app.run(debug=True)

import streamlit as st

import requests

# Title of the Streamlit app

st.title("Data Backup and Recovery System")

# Input fields for bucket and file name

bucket = st.text\_input("Enter Bucket Name", "my-bucket")

file\_name = st.text\_input("Enter File Name to Restore", "data\_backup.csv")

# Display a loading spinner while making the request

with st.spinner("Restoring file..."):

# Restore Button

if st.button("Restore File"):

try:

# Make a request to the Flask API to restore the file

response = requests.get(f"http://localhost:5000/restore/{bucket}/{file\_name}")

# Check if the request was successful

if response.status\_code == 200:

st.success(response.json().get("message", "File restored successfully."))

else:

st.error(f"Error: {response.json().get('error', 'Something went wrong.')}")

except requests.exceptions.RequestException as e:

# Handle exceptions such as network errors or connection issues

st.error(f"Failed to connect to the API: {e}")