|  |
| --- |
|  |

from flask import Flask,redirect,url\_for,render\_template,request

import ibm\_boto3

from ibm\_botocore.client import Config,

COS\_ENDPOINT = "https://s3.jp-tok.cloud-object-storage.appdomain.cloud"

COS\_API\_KEY\_ID = "fDuv-mL8y30SDvGs\_mG-BGjWeSrSYXT2Mj7iNf7BKjQH"

COS\_INSTANCE\_CRN = "crn:v1:bluemix:public:cloud-object-storage:global:a/870cd186abe443a38c10533f71e00157:11e43fa8-e1ad-4236-b1e6-a8385ef2cffe::"

# Create resource https://s3.ap.cloud-object-storage.appdomain.cloud

cos = ibm\_boto3.resource("s3",

ibm\_api\_key\_id=COS\_API\_KEY\_ID,

ibm\_service\_instance\_id=COS\_INSTANCE\_CRN,

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

endpoint\_url=COS\_ENDPOINT

)

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

def get\_item(bucket\_name, item\_name):

print("Retrieving item from bucket: {0}, key: {1}".format(bucket\_name, item\_name))

try:

file = cos.Object(bucket\_name, item\_name).get()

print("File Contents: {0}".format(file["Body"].read()))

except ClientError as be:

print("CLIENT ERROR: {0}\n".format(be))

except Exception as e:

print("Unable to retrieve file contents: {0}".format(e))

def get\_bucket\_contents(bucket\_name):

print("Retrieving bucket contents from: {0}".format(bucket\_name))

try:

files = cos.Bucket(bucket\_name).objects.all()

files\_names = []

for file in files:

files\_names.append(file.key)

print("Item: {0} ({1} bytes).".format(file.key, file.size))

return files\_names

except ClientError as be:

print("CLIENT ERROR: {0}\n".format(be))

except Exception as e:

print("Unable to retrieve bucket contents: {0}".format(e))

def delete\_item(bucket\_name, object\_name):

try:

cos.delete\_object(Bucket=bucket\_name, Key=object\_name)

print("Item: {0} deleted!\n".format(object\_name))

except ClientError as be:

print("CLIENT ERROR: {0}\n".format(be))

except Exception as e:

print("Unable to delete object: {0}".format(e))

def multi\_part\_upload(bucket\_name, item\_name, file\_path):

try:

print("Starting file transfer for {0} to bucket: {1}\n".format(item\_name, bucket\_name))

# set 5 MB chunks

part\_size = 1024 \* 1024 \* 5

# set threadhold to 15 MB

file\_threshold = 1024 \* 1024 \* 15

# set the transfer threshold and chunk size

transfer\_config = ibm\_boto3.s3.transfer.TransferConfig(

multipart\_threshold=file\_threshold,

multipart\_chunksize=part\_size

)

# the upload\_fileobj method will automatically execute a multi-part upload

# in 5 MB chunks for all files over 15 MB

with open(file\_path, "rb") as file\_data:

cos.Object(bucket\_name, item\_name).upload\_fileobj(

Fileobj=file\_data,

Config=transfer\_config

)

print("Transfer for {0} Complete!\n".format(item\_name))

except ClientError as be:

print("CLIENT ERROR: {0}\n".format(be))

except Exception as e:

print("Unable to complete multi-part upload: {0}".format(e))

@app.route('/')

def index():

files = get\_bucket\_contents('flaskapp-by-akaash')

return render\_template('index.html', files=files)

@app.route('/deletefile', methods = ['GET', 'POST'])

def deletefile():

if request.method == 'POST':

bucket=request.form['bucket']

name\_file=request.form['filename']

delete\_item(bucket,name\_file)

return 'file deleted successfully <a href="/"> GO to Home</a>'

if request.method == 'GET':

return render\_template('delete.html')

@app.route('/uploader', methods = ['GET', 'POST'])

def upload():

if request.method == 'POST':

bucket=request.form['bucket']

name\_file=request.form['filename']

f = request.files['file']

multi\_part\_upload(bucket,name\_file,f.filename)

return 'file uploaded successfully <a href="/">GO to Home</a>'

if request.method == 'GET':

return render\_template('upload.html')

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

app.run(host='0.0.0.0',port=8080,debug=True)