**Python Script for Chunked Download and Merge from S3**

This Python script is designed to download large files from an IBM Cloud Object Storage (COS) bucket in chunks and then merge those chunks locally if file size is greater than 2GB. This approach is particularly useful when dealing with large files to avoid timeouts and manage memory efficiently.

**Importing Necessary Modules**

import os

import ibm\_boto3

from ibm\_botocore.client import Config, ClientError

from pathlib import Path

import time

from datetime import datetime

import sys

The script starts by importing necessary libraries:

* os for interacting with the operating system.
* ibm\_boto3 and ibm\_botocore for interacting with IBM Cloud Object Storage.
* Path from pathlib for handling file paths.
* time and datetime for managing timestamps.
* sys for system-specific parameters and functions.

**Constants for IBM COS values**

# Constants for IBM COS values

COS\_ENDPOINT = "https://s3.us-south.cloud-object-storage.appdomain.cloud"

COS\_API\_KEY\_ID = ""

COS\_INSTANCE\_CRN = ""

proxy = "http://vz-proxy.pncint.net:80"

os.environ["https\_proxy"] = proxy

Here, the script sets constants for accessing IBM COS:

* COS\_ENDPOINT: The endpoint URL for the COS service.
* COS\_API\_KEY\_ID: The API key for authenticating to the COS service.
* COS\_INSTANCE\_CRN: The Cloud Resource Name (CRN) for the COS instance.
* proxy: The proxy server to be used for HTTP requests.

The os.environ line sets the HTTPS proxy environment variable.

**Creating the IBM COS Resource**

# Create resource

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

)

The script creates an IBM COS resource using ibm\_boto3.resource, providing necessary authentication details and configuration.

**Function: download\_chunk**

def download\_chunk(bucket\_name, key, start\_byte, end\_byte, part\_number, temp\_file\_path):

range\_header = f'bytes={start\_byte}-{end\_byte}'

response = cos.meta.client.get\_object(Bucket=bucket\_name, Key=key, Range=range\_header)

temp\_file\_chunk\_path = f"{temp\_file\_path}.part{part\_number}"

with open(temp\_file\_chunk\_path, 'wb') as f:

f.write(response['Body'].read())

print(f'Chunk {part\_number} downloaded: {start\_byte} to {end\_byte}')

This function downloads a specific chunk of a file from the COS bucket:

* range\_header specifies the byte range for the chunk.
* get\_object retrieves the specified chunk.
* The chunk is saved to a temporary file part (temp\_file\_chunk\_path).

**Function: merge\_chunks**

def merge\_chunks(temp\_file\_path, local\_file\_path, num\_parts):

with open(local\_file\_path, 'wb') as output\_file:

for part\_number in range(1, num\_parts + 1):

temp\_file\_chunk\_path = f"{temp\_file\_path}.part{part\_number}"

with open(temp\_file\_chunk\_path, 'rb') as chunk\_file:

output\_file.write(chunk\_file.read())

os.remove(temp\_file\_chunk\_path) # Remove the chunk file after merging

print(f"All chunks merged into {local\_file\_path}")

This function merges all the downloaded chunks into a single file:

* It iterates through each part, reads it, and writes it to the final file.
* After merging, it removes the temporary chunk files.

**Function: download\_bucket\_contents**

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

print(f"Retrieving bucket contents from: {bucket\_name}")

try:

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

for file in files:

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

start = time.time()

start\_time = datetime.now()

print(f"Item Start Time = {start\_time}")

print(f"Downloading Item: {file.key} ({file.size} bytes)")

local\_path\_filename = f"/home/pj72963/IDF\_PY/{file.key}"

directory\_path = os.path.dirname(local\_path\_filename)

Path(directory\_path).mkdir(parents=True, exist\_ok=True)

temp\_file\_path = local\_path\_filename + ".temp"

if file.size > 0:

part\_size = 2 \* 1024 \* 1024 \* 1024 # 2GB

part\_number = 1

num\_parts = (file.size + part\_size - 1) // part\_size

for start\_byte in range(0, file.size, part\_size):

end\_byte = min(start\_byte + part\_size - 1, file.size - 1)

download\_chunk(bucket\_name, file.key, start\_byte, end\_byte, part\_number, temp\_file\_path)

part\_number += 1

merge\_chunks(temp\_file\_path, local\_path\_filename, num\_parts)

print("Successfully Downloaded and Merged")

else:

print("Skipping Item because Item size is Zero byte")

end = time.time()

time\_elapsed = end - start

print(f"Item Time\_elapsed = {time\_elapsed}")

end\_time = datetime.now()

print(f"Item End Time = {end\_time}")

except ClientError as be:

print(f"CLIENT ERROR: {be}\n")

except Exception as e:

print(f"Unable to retrieve bucket contents: {e}")

This function retrieves the contents of the specified bucket and processes each file:

* It logs the start time and file details.
* It constructs the local file path and ensures the directory exists.
* If the file size is greater than 0, it downloads the file in 2GB chunks using download\_chunk.
* After all chunks are downloaded, it merges them using merge\_chunks.
* It logs the end time and any errors encountered during the process.

**Example Usage**

# Example usage

download\_bucket\_contents('your-bucket-name')

To use the script, call the download\_bucket\_contents function with the name of the bucket you want to download.

**Summary**

This script efficiently downloads large files from IBM COS by splitting them into manageable chunks and merging them locally. It ensures that large files are handled without timeouts and excessive memory usage, providing a robust solution for large file transfers.