

Uploading Large Files to IBM Cloud Object Storage with Aspera and Python

James Belton
Global IT Architect, IBM Cloud Client Success

December 2020

Introduction

IBM Cloud Object Storage is a means to store massive amounts of data, in an unstructured form, very cheaply. By unstructured, I mean that beyond the 'bucket' construct, there's no structure to the way in which the data is held – so no folders or subfolders.

I'm going to assume here that you already know the basics of IBM Cloud Object Storage – if you don't, then I suggest you read my guide entitled 'An Overview of IBM Cloud Object Storage' which you will find on github here: <https://github.com/jamesbeltonIBM/An-Overview-of-IBM-Cloud-Object-Storage>

As that guide states, there are two main ways to interact with Object Storage. The first is to use the GUI within the IBM Cloud console (at <https://cloud.ibm.com>) but this is only good for fairly small files. The second is via the Object Storage API, which means that you need to write a bit of code and again, I have documents available on github which will get you started, such as the document here: <https://github.com/jamesbeltonIBM/Making-Objects-Publicly-Available-from-IBM-Cloud-Object-Storage->

The problem with uploading larger files is that the upload can take some time to complete, especially over the internet where speeds can vary, especially over long distances. To solve this problem, IBM has enabled another of its technologies – called Aspera – to work with Cloud Object Storage and by using Aspera to transmit data, upload and downloads speeds are both faster and more consistent, even over long distances.

Aspera is file transfer software but unlike ftp, http and similar utilities, it uses proprietary Fasp Transfer Technology. Fasp has been designed to move data up to hundreds of times faster than ftp and http, regardless of the file size, transfer distance or network conditions, so is ideal for use when transferring large files, reliably. You can read more about Aspera at <https://www.ibm.com/cloud/aspera>.

The purpose of this guide is to show you how to use Aspera to transfer data quickly to IBM Cloud Object Storage, via a Python program. This program can be either used at the command line (as we'll do here), or it can be used as part of a wider application.

The program itself leans heavily on the IBM Cloud documentation at <https://cloud.ibm.com/docs/cloud-object-storage?topic=cloud-object-storage-aspera> but I hope this guide will help understand the documentation in more detail and help you avoid the issues that I had to get the code working.

Pre-requisites

You'll essentially need three things to complete this task.

- 1) An IBM Cloud Account
- 2) An Object Storage Bucket
- 3) A computer with Python 2.x installed

IBM Cloud Account

If you don't already have an IBM Cloud account, the hop over to <https://cloud.ibm.com> to create one. I won't go into full details on how to do that here, but if you need help, then you can find details (and lots more besides!) in my YouTube series 'IBM Cloud Foundation Skill Series' at <https://www.youtube.com/playlist?list=PLmesOgYt3nKCfsXqx-A5k1bP7t146U4rz>

An Object Storage Bucket

Once you have an IBM Cloud account (even a 'Lite' account) you'll be able to create an Object Storage service instance and be able to create a bucket. Again, this is outside of the scope of this guide but in general, the process is pretty simple. From you IBM Cloud account, click 'Catalog', click the 'Object Storage' panel and then enter the requisite details on the 'Create' screen. Note that you can use a Lite plan if you wish. Once you have clicked create and your service has provisioned, choose 'Buckets' from the left-hand menu and then hit the 'Create Bucket' button. Follow the wizard.

A computer with Python 2.x installed

Since the program that we're going to put together is coded in Python, you'll need to have Python installed on your computer. Unfortunately, one of the libraries needed does not yet run under Python 3, so we're going to have to stick with Python 2.x.. The version that I have used here is Python 2.7.16, running on macOS.

Again, I'm not going to provide a guide here on how to install Python. Most macOS, Linux and UNIX systems probably already have it installed (to check, open a command prompt and simply type 'python'), but if your system is missing it, then head over to <https://www.python.org/downloads/> and download version 2.7.17 (again, Python3 won't work because the cos-aspera library is not yet available for it).

Some other Python Requirements

There are also some Python libraries that are needed. These are:

- requests
- ibm-cos-sdk
- cos-aspera

To install these, open a terminal and type these commands, allowing each preceding one to complete:

```
python pip -m install requests
python pip -m install ibm-cos-sdk
python pip -m install cos-aspera
```

Information that you will need

There is some other information that you'll need which is used to connect to your Object Storage instance (we're now going to refer to this as 'COS') from your Python code. These are:

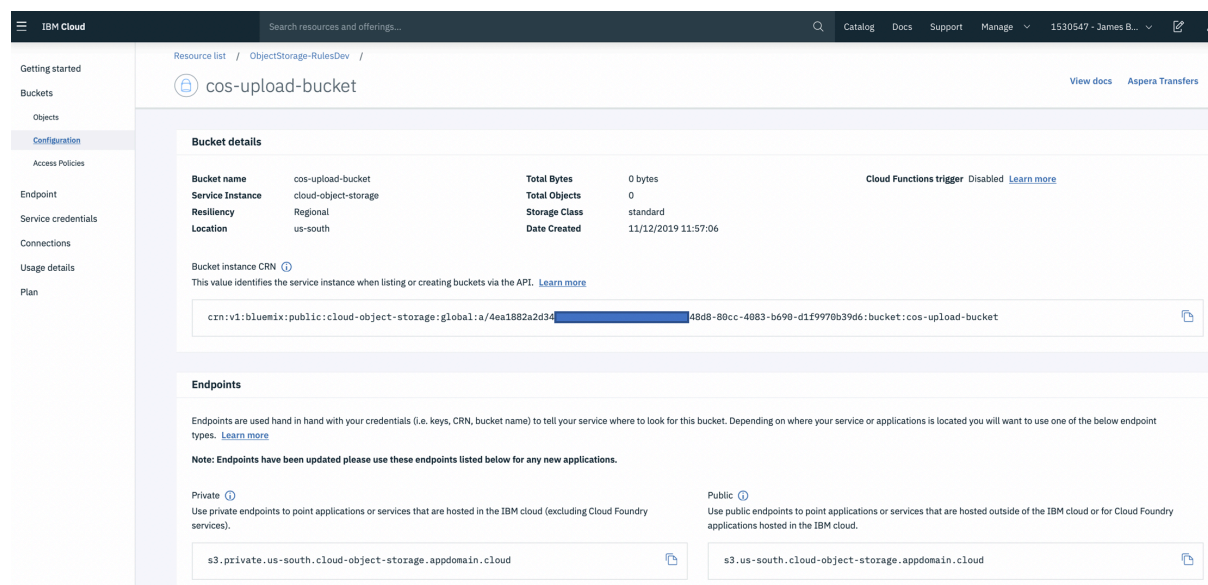
- The COS Endpoint
- The COS Resource CRN
- The COS bucket location
- The COS API Key

These are simple to locate.

The COS Endpoint, COS Resource CRN, COS Bucket Location

This is effectively the web-accessible address to the point where your bucket resides. The easiest way to find them is to access your COS instance from the console at <https://cloud.ibm.com> and then click 'Buckets' from the left-hand menu. You should then see the name of your target bucket – click on that.

Next, click on the Configuration menu item and you should see a screen similar to this one (note the details on the screen will not match this):



For the COS Endpoint, you need the 's3' URL under the Public heading in the Endpoints section of the screen. In this example, the endpoint is:

```
s3.us-south.cloud-object-storage.appdomain.cloud
```

The Resource CRN is shown as the Bucket instance CRN in the Bucket Details section of the screen BUT we need to just change the end of what is shown – removing the bucket:cos-upload-bucket part at the end and simply replacing it with another colon (so that there are two colons on the end of the string – note the 'cos-upload-bucket' part of this is the name of my bucket, so yours may well be different).

We'll end up with a string that looks something like (and yours will have the same format but will have a different value and I have obfuscated mine):

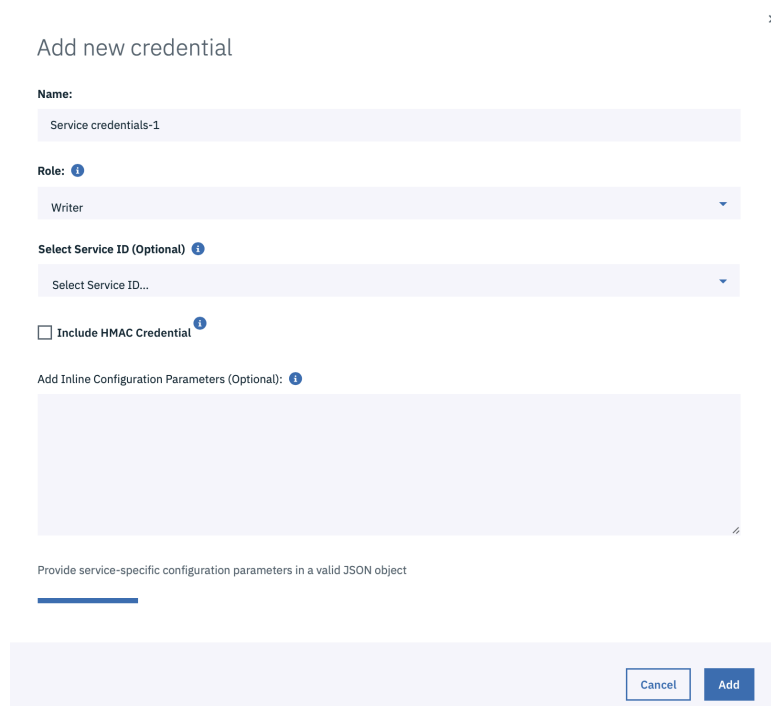
```
crn:v1:bluemix:public:cloud-object-  
storage:global:a/4ea1882a2d3xxxxxxxxxxxxxxxxxxxxxxxx48d8-80cc-4083-  
b690-d1f9970b39d6::
```

The COS Bucket Location is also shown on this screen under Bucket Details. Here, Location is shown as 'us-south', so this is the value that I will use – again, this may be different in your case, it depends how and where you created your bucket.

The COS API Key

The COS API Key is a credential that basically provides your script access to the bucket. Without it, the script will not be allowed to connect. Here, we're going to create a new API Key.

First, click 'Service Credentials' on the left hand menu and then click the 'New Credential' button. A panel like this one will be displayed:



Add new credential

Name: Service credentials-1

Role: ? Writer

Select Service ID (Optional) ? Select Service ID...

☐ Include HMAC Credential ?

Add Inline Configuration Parameters (Optional): ?

Provide service-specific configuration parameters in a valid JSON object

Cancel Add

Provide a name for your Credential in the Name field – use something meaningful, so that you know in future what you created it for.

Under Rule, stick with 'Writer' and then press 'Add'.

It'll take a second or two to create the rule and when it shows in the list, click on View Credentials. You'll see some JSON formatted information, it's the value of "apikey" that you need, for example:

```
python-upload-script-creds 13 DEC 2019 - 10:44:30 AM View credentials
{
  "apikey": "jW3aKm1ZPrIYe1VNkk1CywV1N_T7rdB5c1_M8Nxx_Tqn",
}
```

NOTE: Do not share this information, as it will give someone unauthorised access to your bucket, I will be deleting this key once I've written this tutorial!

We now have the information that we need.

The Python Code

In this section, we'll put the Python code together. A full code listing is available in with this document in the github repository but I'm breaking the code down here so that you can see how it's working.

I've used a text editor (vi) to create my code, which I have saved into a file called cos-upload.py

Block 1 – Import the required Libraries

The first part of the code imports libraries from python libraries which the upload process will use.

```
#Import required Libraries

import sys
import ibm_boto3
from ibm_botocore.client import Config
from ibm_s3transfer.aspera.manager import AsperaTransferManager
from ibm_s3transfer.aspera.manager import AsperaConfig
```

You don't need to change anything here, just make sure that the Python libraries listed in the pre-requisites section are installed. The 'sys' library should already be installed – this will allow us to pass the name of the file to be uploaded and the name we want it to store as in COS from the command line, when we run the program.

Block 2 – Set the Required Variables with Info About Your COS Instance

```
# Set required variables with info about the COS instance

COS_ENDPOINT = "https://s3.us-south.cloud-object-storage.appdomain.cloud"
COS_API_KEY_ID = "jW3aKm1ZPrIYe1VNkk1CywV1N_T7rdB5c1_M8Nxx_Tqn"
COS_AUTH_ENDPOINT = "https://iam.cloud.ibm.com/identity/token"
COS_RESOURCE_CRN = "crn:v1:bluemix:public:cloud-object-storage:global:a/4ea1882a2d3xxxxxxxxxxxxxxxxxxxxxxxxx48d8-80cc-4083-b690-d1f9970b39d6::"
COS_BUCKET_LOCATION = "us-south"
```

This is the only bit of code that you will need to change to get your program working – or rather have it connect to your instance of COS. Simply update these values to reflect those which you gathered in the 'Information You Will Need' section above.

Two things to note:

- 1) You must prefix the COS Endpoint value you collected above with 'https://' and;

- 2) the COS_AUTH_ENDPOINT value will always be as shown above (no need to change that value).

Block 3 – Set up the Client Resource

You don't need to change this bit of code. What it's basically doing is creating a variable object called 'cos' which holds the connection information that the Aspera Transfer Manager software needs to connect to COS. This uses the information provided in the variables set up in Code Block 2.

```
# Create the client resource

cos = ibm_boto3.client("s3",
    ibm_api_key_id=COS_API_KEY_ID,
    ibm_service_instance_id=COS_RESOURCE_CRN,
    ibm_auth_endpoint=COS_AUTH_ENDPOINT,
    config=Config(signature_version="oauth"),
    endpoint_url=COS_ENDPOINT
)
```

Block 4 – Initiate the Transfer Manager Client and Define the Transfer Configuration

Again, you don't need to change this code.

```
#initiate the client

transfer_manager = AsperaTransferManager(cos)

# use multiple sessions/threads for the upload

ms_transfer_config = AsperaConfig(multi_session="all",
    target_rate_mbps=2500,
    multi_session_threshold_mb=100)
```

The first part of this code is initiating the Aspera Transfer Manager, using the client details provided in the object 'cos' that was created in Code Block 3.

The second part of the code set up the sessions or threads that the transfer will use. Here, we're going to use multiple sessions to upload files. We're going to aim to have a total transfer rate of 2,500mbps, with each session having a maximum threshold of 100mb.

You can play with these numbers if you wish but the documentation suggests this is optimal in most situations.

Block 5 – File Upload Information

This block of code tells the program which COS bucket we are targeting, the name of the local file that we are uploading and the name that we want to give the file when it's in COS.

```
# File upload information

bucket_name = "cos-upload-bucket"
upload_filename = sys.argv[1]
object_name = sys.argv[2]

print ("Uploading file to " + bucket_name)
```

Here, the 'bucket_name' value is static but the upload_filename and object_name values are passed from the command line when the program is run.

You could also pass the bucket_name as a variable by changing that line to:

```
bucket_name = sys.argv[3]
```

and passing bucket_name as the third value when running the script from the command line.

The print line is used to output a simple message to the screen when the script is run.

Block 6 – Create the Transfer Manager Process, Perform the Upload and Wait for it to Complete

Again, you don't need to change this code.

```
# Create Transfer manager
with AsperaTransferManager(cos) as transfer_manager:

    # Perform upload
    future = transfer_manager.upload(upload_filename, bucket_name, object_name)

    # Wait for upload to complete
    future.result()

print ("Upload Complete!")
```

The first line initiates the Aspera Transfer Manager and the second line performs the upload using the file name, bucket name and object name passed in Block 5. The third line waits for the result.

When the upload is completed, a message is printed to the screen.

Full Code Listing

This is a full listing of the code used. Note you can download the code file from the same git repository that holds this document. The filename is cos-upload.py

```
#Import required Libraries
```

```
import sys
import ibm_boto3
from ibm_botocore.client import Config
from ibm_s3transfer.aspera.manager import AsperaTransferManager
from ibm_s3transfer.aspera.manager import AsperaConfig
```



```

# Set required variables with info about the COS instance
# Change these values so that they match your environment

COS_ENDPOINT = "https://s3.us-south.cloud-object-storage.appdomain.cloud"
COS_API_KEY_ID = "jW3aKmlZPrIYelVNkklCyWVlN_T7rdB5cl_M8Nxx_Tqn"
COS_AUTH_ENDPOINT = "https://iam.cloud.ibm.com/identity/token"
COS_RESOURCE_CRN = "crn:v1:bluemix:public:cloud-object-
storage:global:a/4ea1882a2d3xxxxxxxxxxxxxxxxxxxxxxxx48d8-80cc-4083-b690-
dlf9970b39d6::"
COS_BUCKET_LOCATION = "us-south"

# Create the client resource

cos = ibm_boto3.client("s3",
    ibm_api_key_id=COS_API_KEY_ID,
    ibm_service_instance_id=COS_RESOURCE_CRN,
    ibm_auth_endpoint=COS_AUTH_ENDPOINT,
    config=Config(signature_version="oauth"),
    endpoint_url=COS_ENDPOINT
)

#initiate the client

transfer_manager = AsperaTransferManager(cos)

# use multiple sessions/threads for the upload

ms_transfer_config = AsperaConfig(multi_session="all",
    target_rate_mbps=2500,
    multi_session_threshold_mb=100)

# File upload information

bucket_name = "cos-upload-bucket"
upload_filename = sys.argv[1]
object_name = sys.argv[2]

print ("Uploading file to " + bucket_name)

# Create Transfer manager
with AsperaTransferManager(cos) as transfer_manager:

    # Perform upload
    future = transfer_manager.upload(upload_filename, bucket_name,
    object_name)

    # Wait for upload to complete
    future.result()

print ("Upload Complete!")

```

Running the Code

The code can be run from the command line in the following way:

```
python cos-upload.py <</path/to/MyFile.iso>> <<uploaded_obj_name>>
```

For example, to upload a file called 'linux-distro.iso' in the local directory '/usr/local/downloads' and have it stored as an object called 'default-linux-distro.iso', the program would be run as follows:

```
python cos-upload.py /usr/local/downloads/linux-distro.iso default-  
linux-distro.iso
```

Next Steps

This program allows the upload of single files via Aspera to IBM Cloud Object Storage. The next step is to create a full utility which also downloads single files as well as uploads and downloads entire directories.

I'll look to expand the script to provide that functionality in the next iteration.