s3 using Python libraries

This documentation focuses on **access to s3 using Python libraries**, specifically on the use of **boto3** library, and includes a code sample, split into various steps.

Step 0: Install package in Python environment

In order to run the following example you need to have the following packages in your environment:

- **boto3**

You can check this documentation for Install package in Python environment and handle python environments for reproducibility.

Now that you have an environment with the required packages and Python version, we can move to prepare and run the actual code.

Step 1: Configure client

Depending on whether the bucket you want to access is private (needs credentials) or public (no credentials needed), choose one of the following..

To access private buckets that require S3 credentials

Make sure you have both Python 3 and the access keys to your S3 bucket ready. Typically you'll find your access key and secret key in Morpheus under *Tools -> Cypher*.

Start by declaring some initial values for boto3 to know where your bucket is located at. Feel free to copy paste this segment and fill in with your own values.

If you're connecting to buckets hosted at the EUMETSAT side of the European Weather Cloud, the endpoint is: `https://s3.waw3-1.cloudferro.com`

If you're connecting to buckets hosted at the ECMWF side of the European Weather Cloud, the endpoints are:

- CCI1 cluster : https://object-store.os-api.cci1.ecmwf.int
- CCI2 cluster : https://object-store.os-api.cci2.ecmwf.int

```
import os
import io
import boto3


# Initializing variables for the client
S3_BUCKET_NAME = "MyFancyBucket123"  #Fill this in
S3_ACCESS_KEY = "123asdf"  #Fill this in
S3_SECRET_ACCESS_KEY = "123asdf111"  #Fill this in
S3_ENDPOINT_URL = "https://my-s3-endpoint.com"  #Fill this in
```

Lets start by initializing the S3 client with our access keys and endpoint:

```
# Initialize the S3 client
s3 = boto3.client(
        's3',
        endpoint_url=S3_ENDPOINT_URL,
    aws_access_key_id=S3_ACCESS_KEY,
    aws_secret_access_key=S3_SECRET_ACCESS_KEY
)
```

To access public buckets (no credentials required)

```
import os
import io
import boto3

from botocore import UNSIGNED
from botocore.config import Config

# Initializing variables for the client
S3_BUCKET_NAME = "MyFancyBucket123"  #Fill this in
S3_ENDPOINT_URL = "https://my-s3-endpoint.com"  #Fill this in
```

Lets start by initializing the S3 client with our access keys and endpoint:

```
# Initialize the S3 client
s3 = boto3.client(
        's3',
        endpoint_url=S3_ENDPOINT_URL,
    config=Config(
                signature_version=UNSIGNED
        ))
```

Step 2: Perform actions

Case 1: List objects

As a first step, and to confirm we have successfully connected, lets list the objects inside our bucket (up to a 1.000 objects).

```
# List the objects in our bucket
response = s3.list_objects(Bucket=S3_BUCKET_NAME)
for item in response['Contents']:
    print(item['Key'])
```

If you'd want to list *more* than 1000 objects in a bucket, you can use *paginator:*

```
# List objects with paginator (not constrained to a 1000 objects)
paginator = s3.get_paginator('list_objects_v2')
pages = paginator.paginate(Bucket=S3_BUCKET_NAME)

# Lets store the names of our objects inside a list
objects = []
for page in pages:
    for obj in page['Contents']:
        objects.append(obj["Key"])

print('Number of objects: ', len(objects))
```

Where an *obj* looks like this:

```
{'Key': 'MyFile.txt', 'LastModified': datetime.datetime(2021, 11, 11, 0, 39, 23, 320000, tzinfo=tzlocal()),
'ETag': '"2e22f62675cea3445f7e24818a4f6ba0d6-1"', 'Size': 1013, 'StorageClass': 'STANDARD'}
```

Case 2: Read objects

Into memory

Now lets try to read a file from a bucket into Python's memory, so we can work with it inside Python without ever saving the file to our local computer:

```
#Read a file into Python's memory and open it as a string
FILENAME = '/folder1/folder2/myfile.txt'  #Fill this in
obj = s3.get_object(Bucket=S3_BUCKET_NAME, Key=FILENAME)
myObject = obj['Body'].read().decode('utf-8')
print(myObject)
```

Download objects to a file

But *if* you'd want to download the file instead of reading it into memory (e.g. so you can use it with other libraries or applications that expect files), here's how you'd do that:

```
# Downloading a file from the bucket
with open('myfile', 'wb') as f:  #Fill this in
    s3.download_fileobj(S3_BUCKET_NAME, 'myfile', f)
```

Case 3: Upload objects

And similarly you can upload files to the bucket (given that you have write access to the bucket):

```
# Uploading a file to the bucket (make sure you have write access)
response = s3.upload_file('myfile', S3_BUCKET_NAME, 'myfile')  #Fill this in
```

Case 4: Create a bucket

And lastly, creating a bucket (this could take some time):

```
s3.create_bucket(Bucket="MyBucket")
```

Exercise for the reader: you can test the following actions on the bucket (described above):

- upload a file into the new bucket (case 3)
- list the contents of the bucket to verify your file is there (case 1)
- download the file you uploaded (case 2)

Other Resources

- Check a more detailed view into boto3's functionality *(although it does emphasize on Amazon Web Services specifically, you can take a look at the Python code involved)*: https://dashbird.io/blog/boto3-aws-python/
- Check out a full code example at the official boto3 website: https://boto3.amazonaws.com/v1/documentation/api/latest/guide/s3-examples.html
- Check a differently styled tutorial at https://towardsdatascience.com/introduction-to-pythons-boto3-c5ac2a86bb63