

IBM Cloud

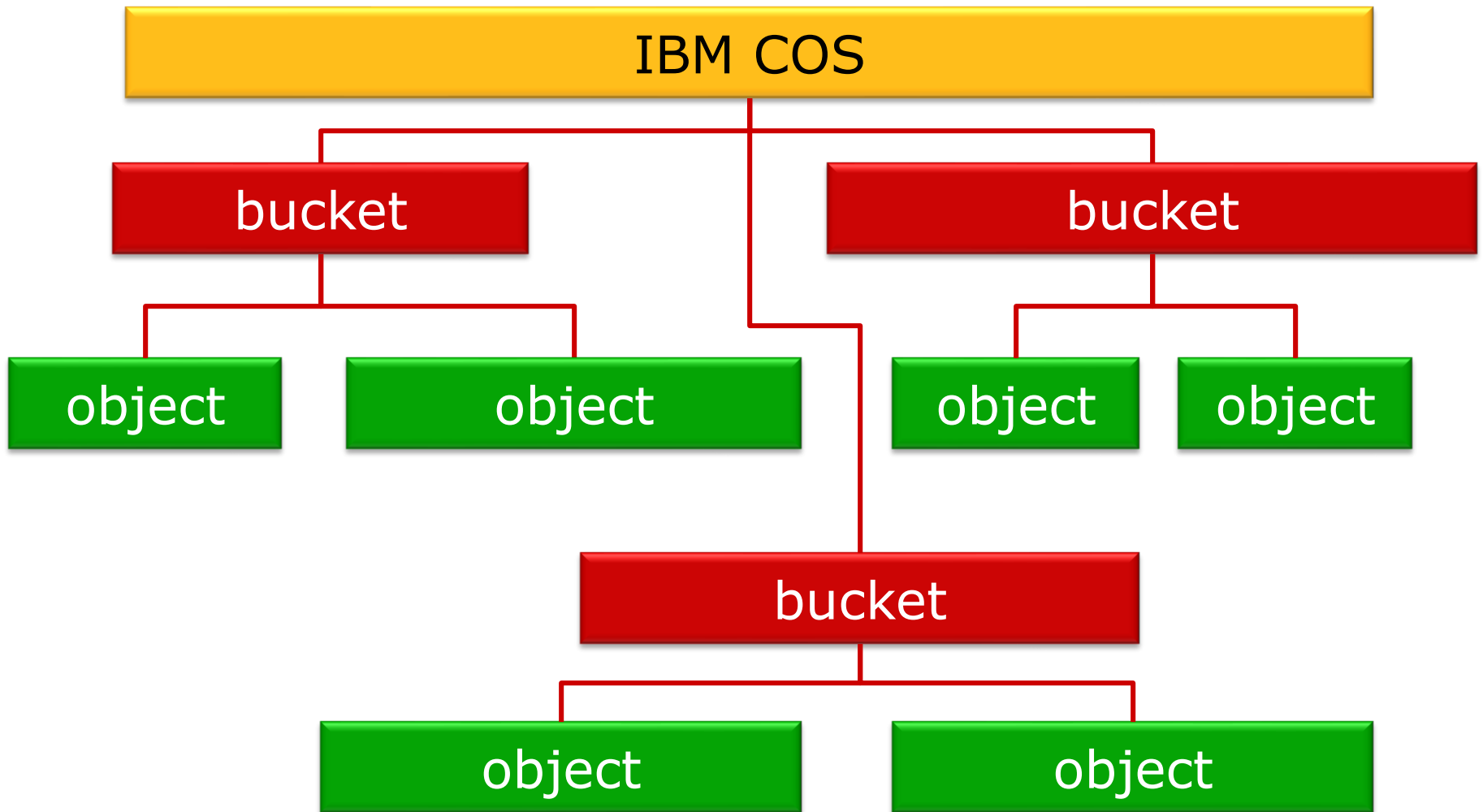
IBM Cloud Object Storage (COS) introduction

Josep Sampé

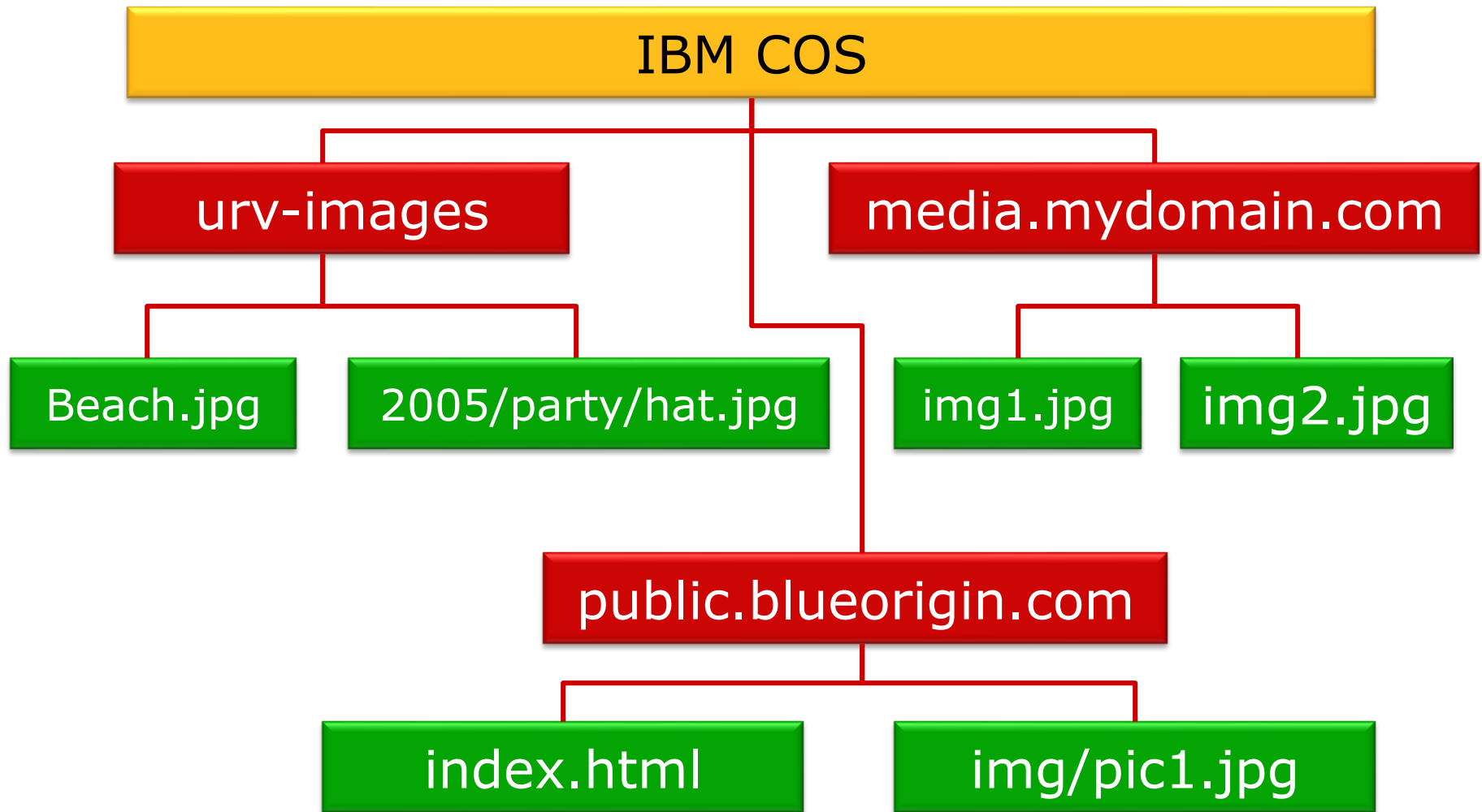
IBM COS

- ❑ It is commonly used for data archiving and backup, web and mobile applications, and as scalable, persistent storage for analytics.
- ❑ Write, read, delete objects 0byte-10TB, single PUT < 10TB
- ❑ Namespace: buckets, keys, objects
- ❑ Accessible using URLs

IBM COS namespace



IBM COS namespace



Access control

- Access log
- Objects are private to the user account
 - Authentication
- Authorization
 - ACL: IBM Cloud users, users identified by email, any user ...
- Digital signature to ensure integrity
- Encrypted access: https

Access methods

☐ HTTP RESTfull API:

- A RESTful API is an application program interface (API) that uses HTTP requests to

- ☐ PUT

- ☐ GET

- ☐ POST

- ☐ HEAD

- ☐ DELETE

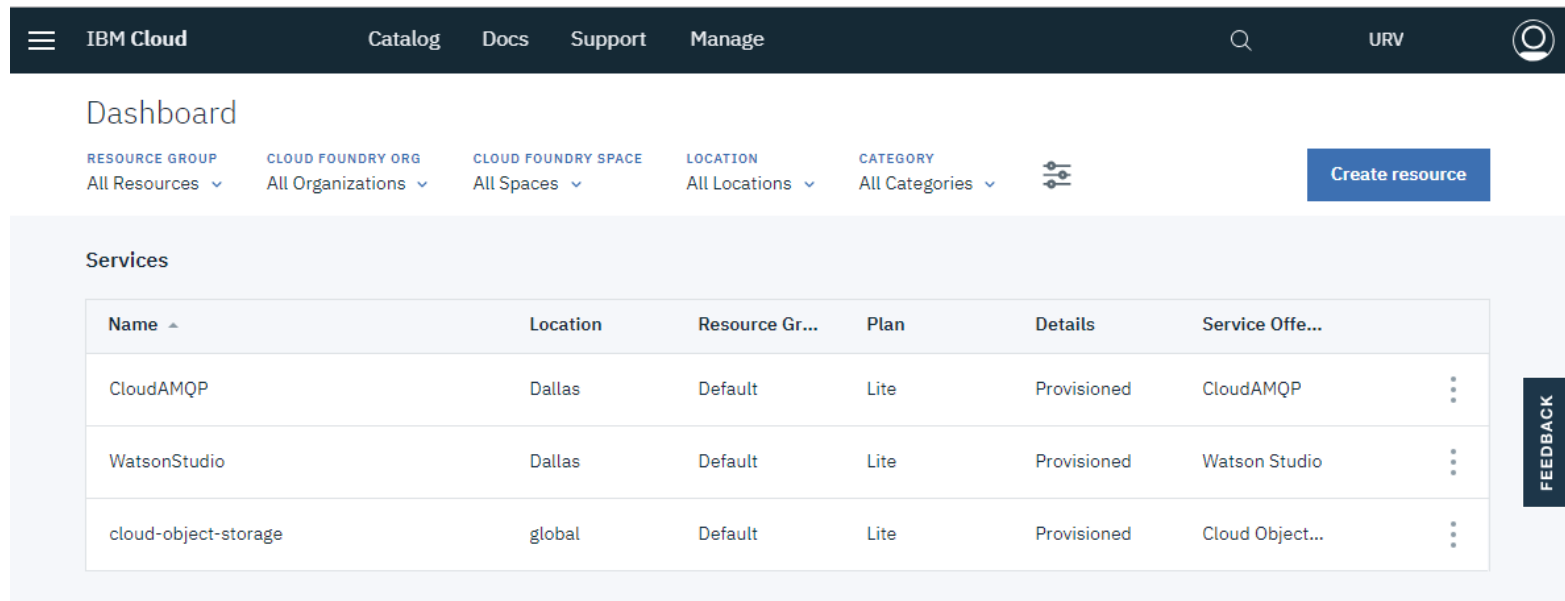
data in the Object storage service.

Account setup

- ❑ Log-in with your IBM Cloud account: create one if you do not yet have one by clicking the sign-up link or by directly navigating to <https://cloud.ibm.com/registration> to get IBM Cloud account
- ❑ To use IBM Cloud Object Storage proceed as follows: open a browser window and navigate to: <https://console.bluemix.net/catalog/services/cloud-object-storage>
- ❑ Click “Create” to get a free instance of the COS Service.

Account setup

- ❑ Navigate to your dashboard by the following url:
<https://console.bluemix.net/dashboard/apps>



The screenshot shows the IBM Cloud Dashboard interface. At the top is a dark navigation bar with links for IBM Cloud, Catalog, Docs, Support, and Manage, along with a search icon, a user profile icon (URV), and a settings icon. Below the navigation bar is a 'Dashboard' section with filters for Resource Group (All Resources), Cloud Foundry Org (All Organizations), Cloud Foundry Space (All Spaces), Location (All Locations), and Category (All Categories). A 'Create resource' button is on the right. The main content area is titled 'Services' and contains a table with the following data:

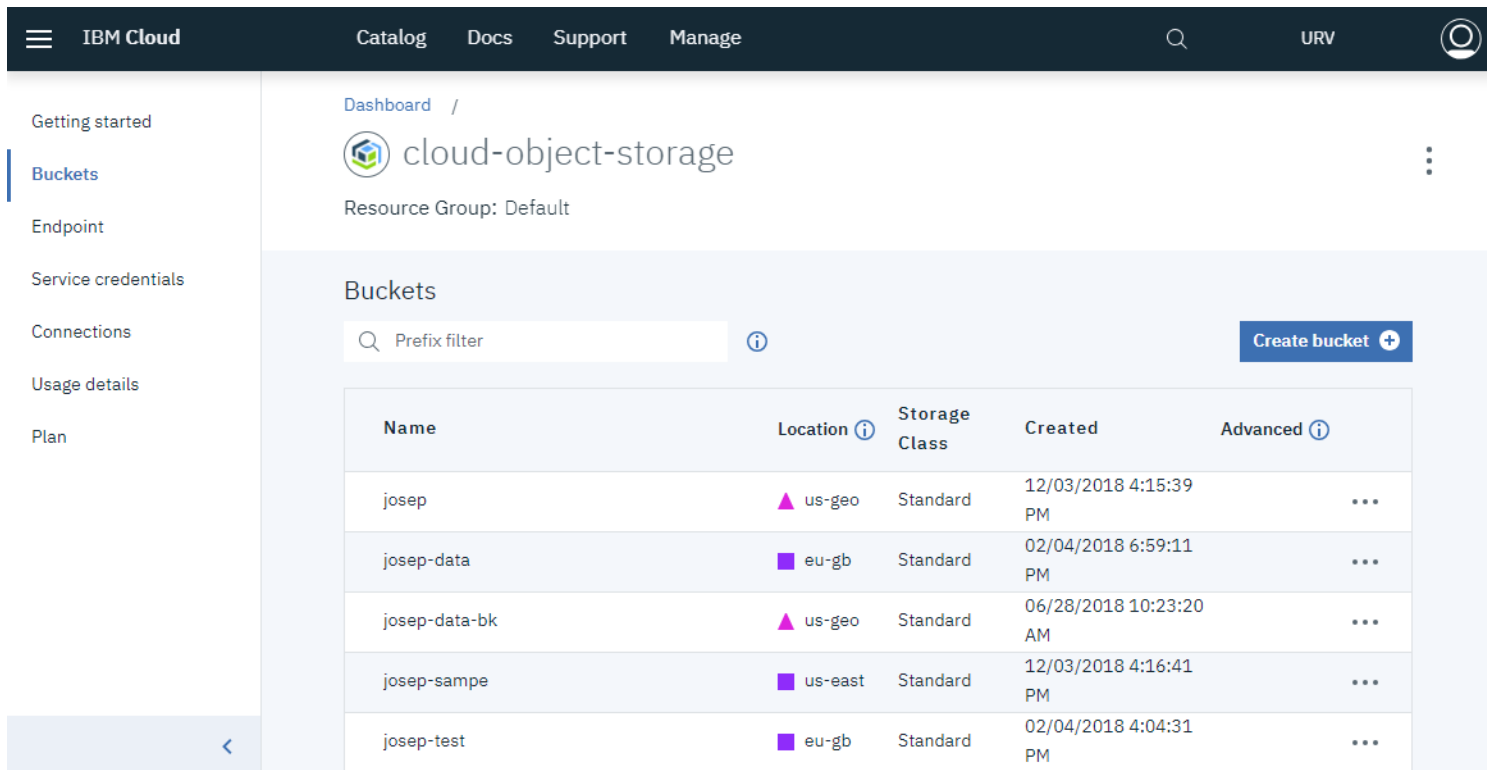
Name ^	Location	Resource Gr...	Plan	Details	Service Offe...
CloudAMQP	Dallas	Default	Lite	Provisioned	CloudAMQP
WatsonStudio	Dallas	Default	Lite	Provisioned	Watson Studio
cloud-object-storage	global	Default	Lite	Provisioned	Cloud Object...

A 'FEEDBACK' button is located on the right side of the table.

- ❑ Click on “cloud-object-storage” to access.

Access methods

□ Web interface



The screenshot displays the IBM Cloud web interface for the 'cloud-object-storage' service. The top navigation bar includes 'IBM Cloud', 'Catalog', 'Docs', 'Support', 'Manage', a search icon, 'URV', and a user profile icon. A left sidebar lists navigation options: 'Getting started', 'Buckets' (selected), 'Endpoint', 'Service credentials', 'Connections', 'Usage details', and 'Plan'. The main content area shows the 'cloud-object-storage' dashboard for the 'Default' Resource Group. It features a 'Buckets' section with a 'Prefix filter' input and a 'Create bucket' button. Below this is a table listing five buckets with their names, locations, storage classes, creation times, and advanced options.

Name	Location ⓘ	Storage Class	Created	Advanced ⓘ
josep	▲ us-geo	Standard	12/03/2018 4:15:39 PM	...
josep-data	■ eu-gb	Standard	02/04/2018 6:59:11 PM	...
josep-data-bk	▲ us-geo	Standard	06/28/2018 10:23:20 AM	...
josep-sampe	■ us-east	Standard	12/03/2018 4:16:41 PM	...
josep-test	■ eu-gb	Standard	02/04/2018 4:04:31 PM	...

Access methods

□ Command line: **aws** package

- *pip3 install awscli --upgrade --user*

```
aws --endpoint-url=http://s3-api.us-geo.objectstorage.softlayer.net  
s3 ls s3://urv-images
```

□ Programming Interface

- E.g., `ibm_boto3` python library
- *pip install -U ibm-cos-sdk --upgrade --user*

Accessing objects

□ **Bucket:** urv-images, **key:** jpg1,

object: a jpg image

■ accessible with:

<https://s3.us-east.objectstorage.softlayer.net/urv-images/jpg1>

□ Pseudo-folders

■ accessible with:

<https://s3.us-east.objectstorage.softlayer.net/urv-images/january/jpg1>

Access control

□ Get the service credentials:

The screenshot shows the IBM Cloud console interface. The left sidebar contains a menu with items: Getting started, Buckets, Endpoint, **Service credentials** (highlighted with a red box), Connections, Usage details, and Plan. The main content area is titled 'cloud-object-storage' and shows 'Resource Group: Default'. Below this, there is a section titled 'Service credentials' with a description: 'Credentials are provided in JSON format. The JSON snippet lists credentials, such as the API key and secret, as well as connection information for the service.' A 'Learn more' link is present. Below the description is a table with the following structure:

KEY NAME	DATE CREATED	ACTIONS
<input type="checkbox"/> WDP-Editor-pywren-donotdelete-pr-5tyfndlx3uyrly	DEC 22, 2018 - 06:25:39 PM	View credentials

The 'New credential +' button is located in the top right corner of the table area and is highlighted with a red box.

To generate HMAC credentials, specify the following in the Add Inline Configuration Parameters (Optional) field: `{"HMAC":true}`

Examples

❑ PUT an object:

```
def put_object(self, bucket_name, key, data):  
    """  
    Put an object in COS. Override the object if the key already exists.  
    :param key: key of the object.  
    :param data: data of the object  
    :type data: str/bytes  
    :return: None  
    """  
    try:  
        res = self.cos_client.put_object(Bucket=bucket_name, Key=key, Body=data)  
        status = 'OK' if res['ResponseMetadata']['HTTPStatusCode'] == 200 else 'Error'  
        try:  
            print('PUT Object {} - Size: {} - {}'.format(key, sizeof_fmt(len(data)), status))  
        except:  
            print('PUT Object {} {}'.format(key, status))  
    except ibm_botocore.exceptions.ClientError as e:  
        raise e
```

Examples

□ GET an object:

```
def get_object(self, bucket_name, key, stream=False, extra_get_args={}):  
    """  
    Get object from COS with a key. Throws StorageNoSuchKeyError if the given key does not exist.  
    :param key: key of the object  
    :return: Data of the object  
    :rtype: str/bytes  
    """  
    try:  
        r = self.cos_client.get_object(Bucket=bucket_name, Key=key, **extra_get_args)  
        if stream:  
            data = r['Body']  
        else:  
            data = r['Body'].read()  
        return data  
    except ibm_botocore.exceptions.ClientError as e:  
        raise e
```

Examples

□ HEAD an object:

```
def head_object(self, bucket_name, key):  
    """  
    Head object from COS with a key. Throws StorageNoSuchKeyError if the given key does not exist.  
    :param key: key of the object  
    :return: Data of the object  
    :rtype: str/bytes  
    """  
    try:  
        metadata = self.cos_client.head_object(Bucket=bucket_name, Key=key)  
        return metadata['ResponseMetadata']['HTTPHeaders']  
    except ibm_botocore.exceptions.ClientError as e:  
        raise e
```

□ DELETE an object:

```
def delete_object(self, bucket_name, key):  
    """  
    Delete an object from storage.  
    :param bucket: bucket name  
    :param key: data key  
    """  
    return self.cos_client.delete_object(Bucket=bucket_name, Key=key)
```


Examples

□ List objects of a bucket:

```
def list_objects(self, bucket_name, prefix=None):
    paginator = self.cos_client.get_paginator('list_objects_v2')
    try:
        if (prefix is not None):
            page_iterator = paginator.paginate(Bucket=bucket_name, Prefix=prefix)
        else:
            page_iterator = paginator.paginate(Bucket=bucket_name)

        object_list = []
        for page in page_iterator:
            if 'Contents' in page:
                for item in page['Contents']:
                    object_list.append(item)
        return object_list
    except ibm_botocore.exceptions.ClientError as e:
        raise e
```

Task: Let's work!

- ❑ Develop a simple Python module called **cos_backend.py** that implements all the previous methods. It will provide access to the IBM COS account.
- ❑ Create a **test.py** file, instantiate the `CosBackend()` class, and make the next operations against COS:
 1. Read a local file and upload it to COS.
 2. Download the file from COS and store it locally with another name.
 3. Make a head request to print the metadata. Concretely to know the object size.
 4. Delete the object.