IBM Cloud

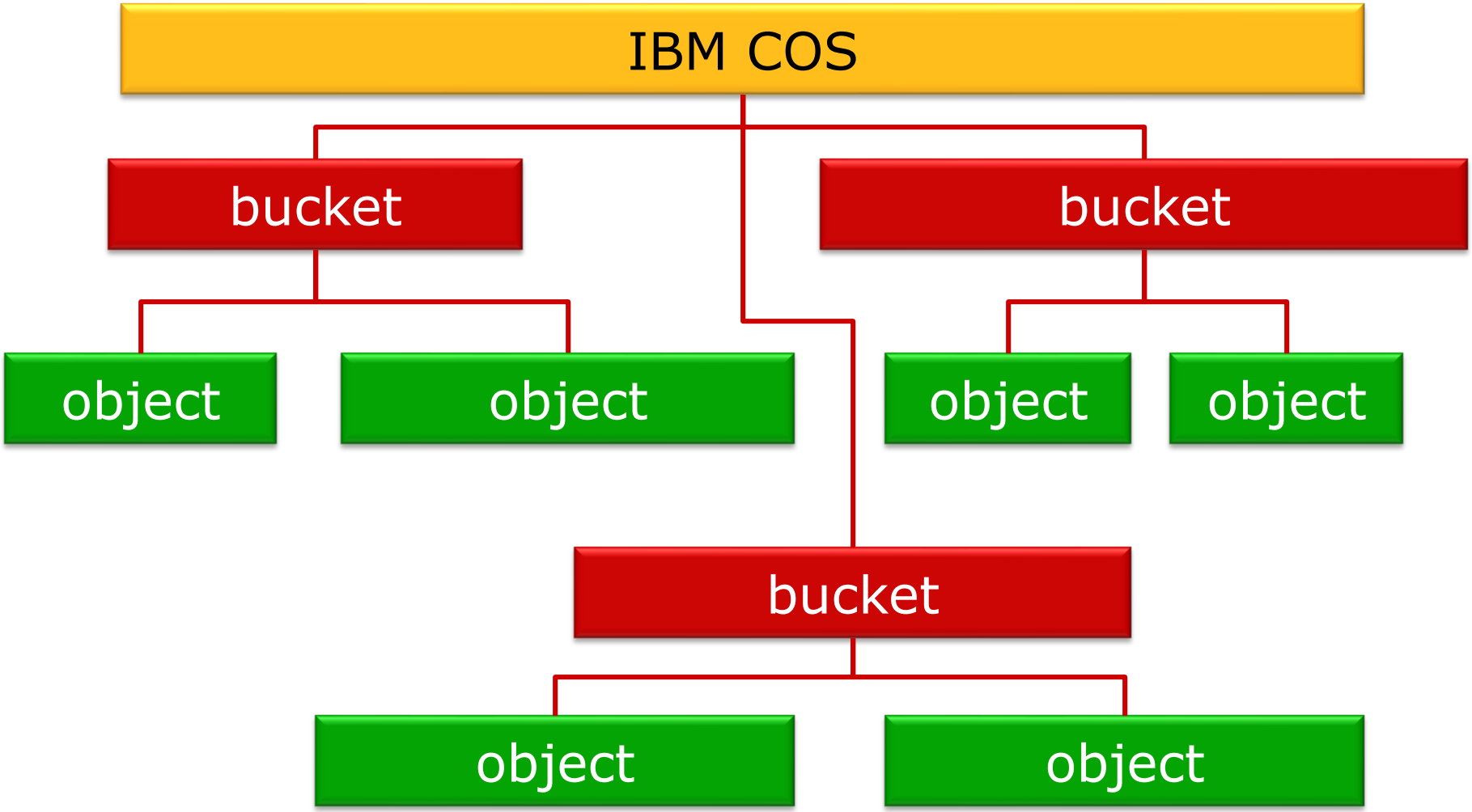
IBM Cloud Object Storage (COS) introduction

Josep Sampé

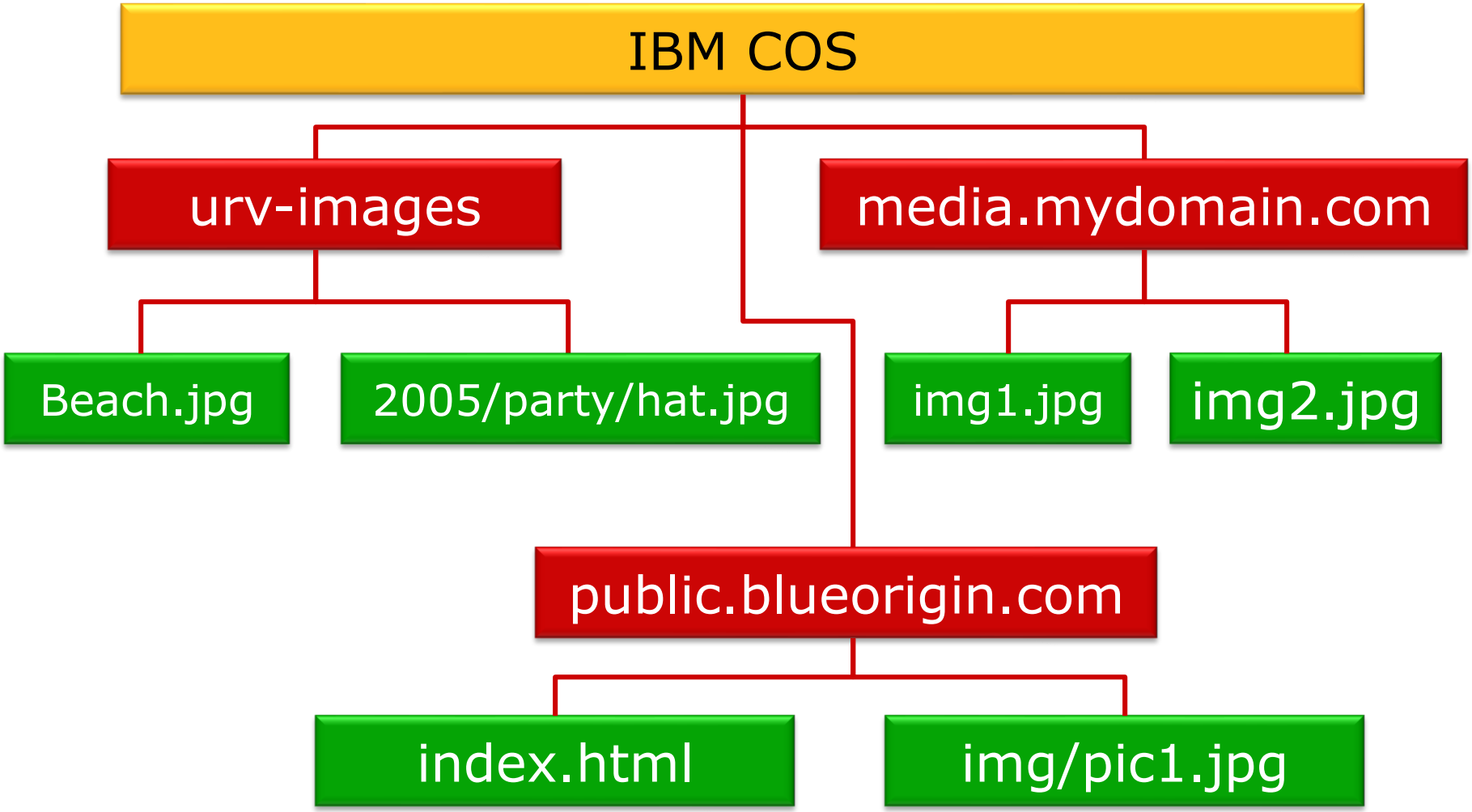
IBM COS

- 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

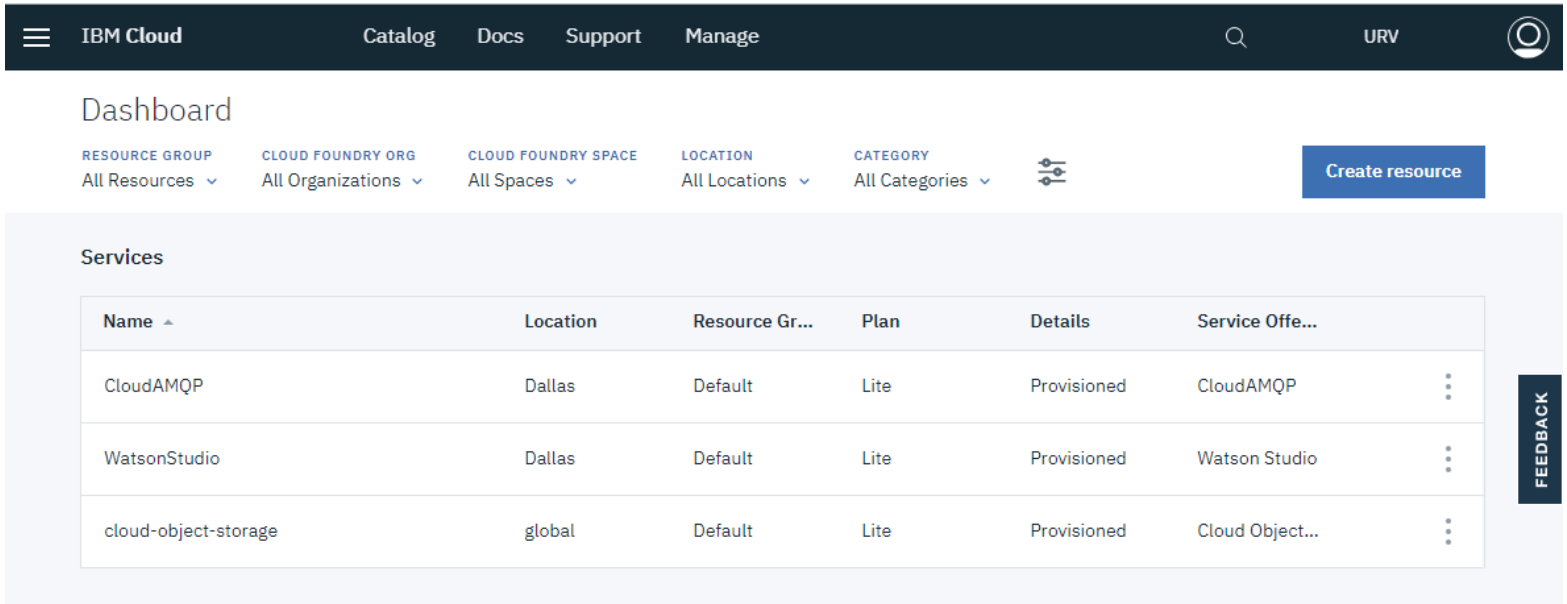
- Log-in with your IBM Cloud account: create one if you do not yet have one by clicking the sign-up link or by directly navigating to https://cloud.ibm.com/registration to get IBM Cloud account

- 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

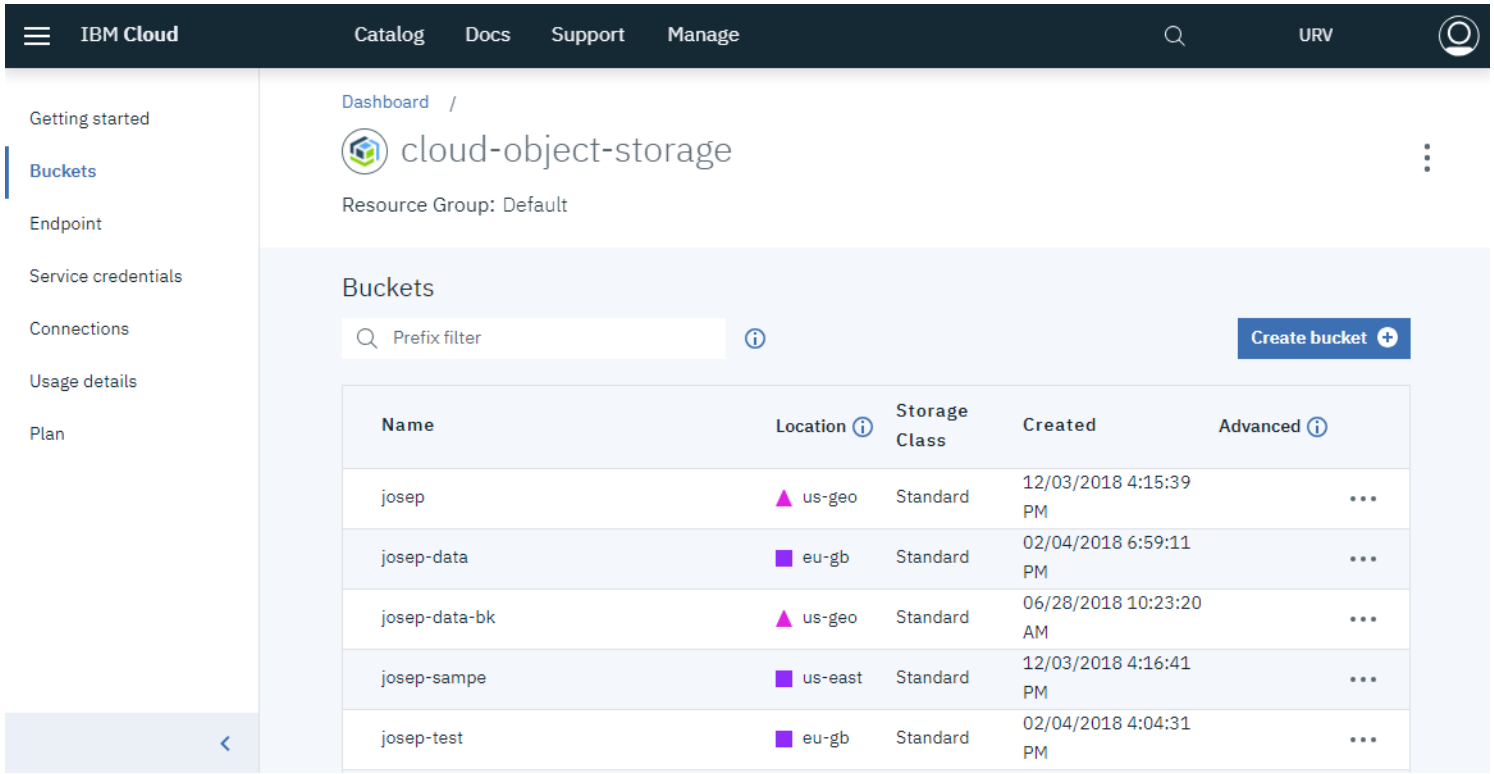
- Navigate to your dashboard by the following url: https://console.bluemix.net/dashboard/apps



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

Access methods

- Web interface



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**

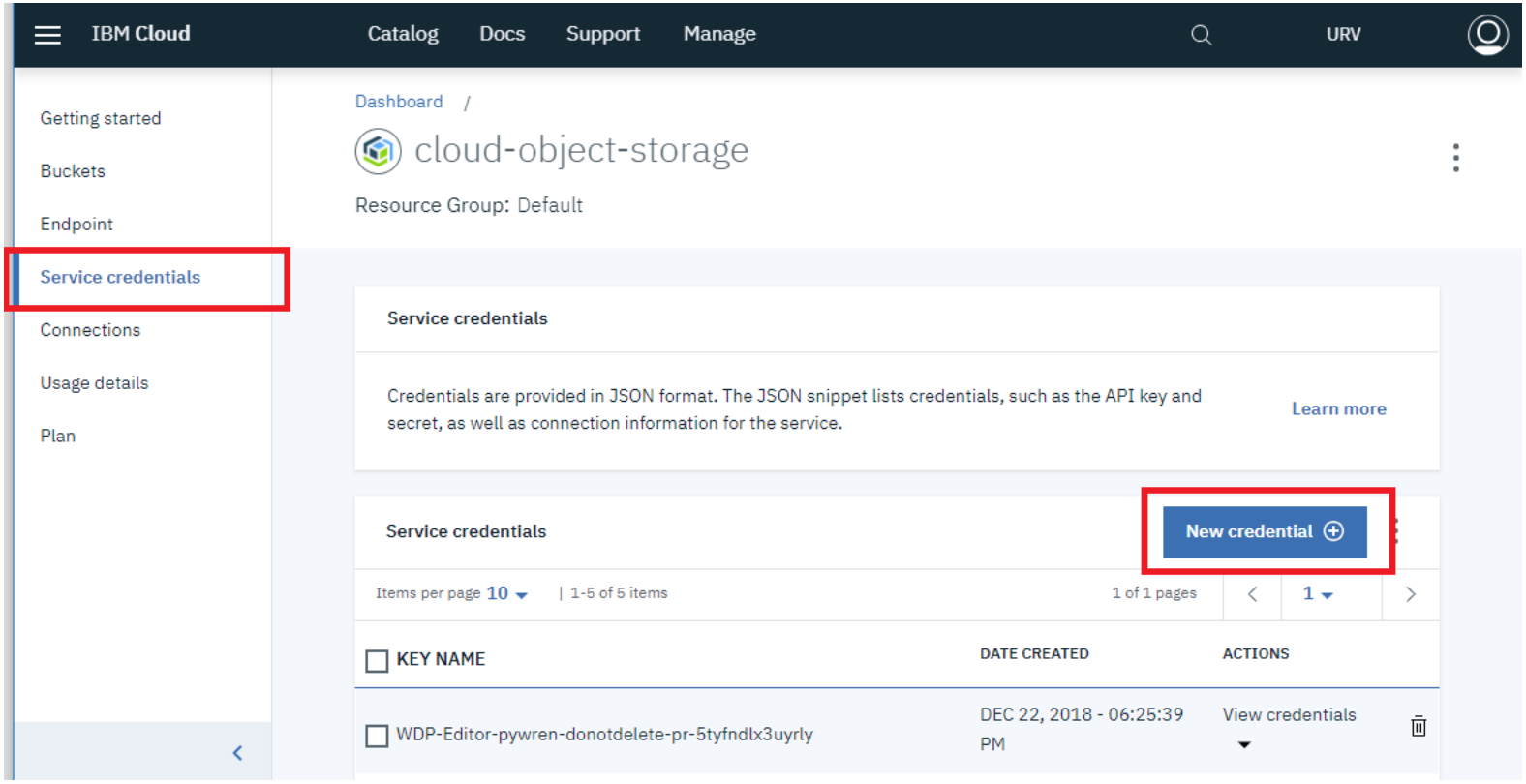
- Pseudo-folders

  - accessible with:

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

Access control

- Get the service credentials:



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

Examples

- Create a COS client:

```
import ibm_boto3
import ibm_botocore


class COSBackend:
    """
    A wrap-up around COS ibm_boto3 APIs.
    """

    def __init__(self, cos_config):
        service_endpoint = cos_config.get('endpoint').replace('http:', 'https:')
        secret_key = cos_config.get('secret_key')
        access_key = cos_config.get('access_key')
        client_config = ibm_botocore.client.Config(max_pool_connections=200,
                                                   user_agent_extra='pywren-ibm-cloud')
        self.cos_client = ibm_boto3.client('s3',
                                           aws_access_key_id=access_key,
                                           aws_secret_access_key=secret_key,
                                           config=client_config,
                                           endpoint_url=service_endpoint)
```

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:

```python
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:

```python
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:

```python
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.