spongeblob.py Documentation

Ayush Goyal

Sep 14, 2018

Contents

1 Overview **1**

2 Installation **3**

3 Example **5**

4 Contents **7**

4.1 Usage 7

CHAPTER 1

Overview

Spongeblob is a library for providing a simple and consistent interface for cloud storage services. Currently, the project supports AWS Simple Storage Service(S3) and Windows Azure blob storage (WABS).

It wraps *boto* for s3 client and *azure-storage* for wabs, and provides a set of basic operations that are commonly used.

CHAPTER 2

Installation

You can fetch spongeblob from pypi via pip.

```
pip install spongeblob
```

CHAPTER 3

Example

To setup a spongeblob client use the following code:

```
import spongeblob

s3 = spongeblob.setup_storage('s3',
                              aws_key='access_key_id',
                              aws_secret='access_key_secret',
                              bucket_name='testbucket')

s3.download_file('/path/to/key', '/path/on/disk')
```

Both these clients implement methods of the spongeblob.storage class.

CHAPTER 4

Contents

4.1 Usage

4.1.1 Setting Up Storage Client

setup_storage is function to initialize a storage client for required cloud storage provider. Refer example on how to create s3/wabs client respectively.

spongeblob.**setup_storage**(*storage_provider*, **args*, ***kwargs*)
Function to setup a Storage object for specified storage provider

Parameters

- **storage_provider** (*str*) – Setup storage with specified storage provider. Supported storage_provider are 's3' and 'wabs'.
- ****kwargs** – For the storage provider used, you will also be required to pass initialization parameters of the respective storage class.

Returns An object for the specified storage class setup with passed storage creds

Return type *S3*, *WABS*

Example

```
from spongeblob import setup_storage

s3 = setup_storage('s3',
                   aws_key='access_key_id',
                   aws_secret='access_key_secret',
                   bucket_name='testbucket')

wabs = setup_storage('wabs',
                     account_name='testaccount',
                     container_name='testcontainer',
                     sas_token='testtoken')
```

4.1.2 S3

S3 class implements the *spongeblob.storage.storage.Storage* class interface documented below for AWS Simple Storage Service. You can use the parameters documented here for a s3 client initialization via *setup_storage()*

class spongeblob.storage.s3.**S3**(*aws_key*, *aws_secret*, *bucket_name*, *boto_config=None*)
A class for managing objects on AWS S3. It implements the interface of Storage base class

__init__(*aws_key*, *aws_secret*, *bucket_name*, *boto_config=None*)
Setup a S3 storage client object

Parameters

- **aws_key** (*str*) – AWS key for the S3 bucket
- **aws_secret** (*str*) – AWS secret for the S3 bucket
- **bucket_name** (*str*) – AWS S3 bucket name to connect to
- **boto_config** (*botocore.client.Config*) – Expects a botocore.client.Config object for boto s3 client connection configuration

4.1.3 WABS

WABS class implements the *spongeblob.storage.storage.Storage* class interface documented below for Windows Azure Blob Storage service. You can use the parameters documented here for a wabs client initialization via `setup_storage()`

class spongeblob.storage.wabs.**WABS**(*account_name*, *container_name*, *sas_token*)
A class for managing objects on Windows Azure Blob Storage. It implements the interface of Storage base class

__init__(*account_name*, *container_name*, *sas_token*)
Setup a Windows azure blob storage client object

Parameters

- **account_name** (*str*) – Azure blob storage account name for connection
- **container_name** (*str*) – Name of container to be accessed in the account
- **sas_token** (*str*) – Shared access signature token for access

4.1.4 Storage API

All supported storage providers in spongeblob implement the storage class for interface. Refer this document for the API available for *spongeblob.storage.s3.S3* and *spongeblob.storage.wabs.WABS* classes

class spongeblob.storage.storage.**Storage**
This is the base class for spongeblob. It defines an interface to be implemented by various storages

copy_from_key(*source_key*, *destination_key*, *metadata=None*)
Copy an object from one key to another key on server side

Parameters

- **source_key** (*str*) – Source key for the object to be copied
- **destination_key** (*str*) – Destination key to store object
- **metadata** (*dict*) – Metadata to be stored along with object

Returns Nothing

Return type None

delete_key(*destination_key*)
Delete an object

Parameters **destination_key** (*str*) – Destination key for the object to be deleted

Returns Nothing

Return type None

download_file(*source_key*, *destination_file*)
Download an object to local filesystem

Parameters

- **source_key** (*str*) – Key for object to be downloaded
- **destination_file** (*str*) – Path on local filesystem to download file

Returns Nothing

Return type None

get_object_properties(*key*, *metadata=False*)
Fetch object properties and optionally metadata as specified by the key. If the object is not found return None.

Parameters

- **key** (*str*) – Key for object for which you want to fetch metdata and properties
- **metadata** (*bool*) – If set to True, metadata will be fetched, else not.

Returns

A dictionary object with some basic properties and object metadata. The returned dict will look like this:

```
{"key": "/key/for/object",
 "size": <size_of_object_in_bytes>,
 "last_modified": <last_modified_timestamp_in_cloud_storage>,
 "metadata": Union(<metadata_dict_of_key>, None)}
```

Return type dict

list_object_keys(*prefix=''*, *metadata=False*, *pagesize=1000*)
List files for the specified prefix. Fetch metdata if set to true

Parameters

- **prefix** (*str*) – String to match when searching files
- **metadata** (*bool*) – If set to True, metadata will be fetched, else not.
- **pagesize** (*int*) – Limits the number of objects fetched in a single api call

Returns

A generator of dict describing objects found by api. The returned dict will look like this

```
{"key": "/key/for/object",
 "size": <size_of_object_in_bytes>,
 "last_modified": <last_modified_timestamp_in_cloud_storage>,
 "metadata": Union(<metadata_dict_of_key>, None)}
```

Metadata key in dict above will be set to `None` if metadata param is False, else metadata will be a dict if metadata is set to True, it will be empty dict if there is no metadata.

Return type Iterator[dict]

list_object_keys_flat (**args*, ***kwargs*)
Takes arguments of list_object_keys function and returns a list of objects instead of generator. This function is retriable unlike list_object_keys.

WARN: Unlike list_object_keys, this function is not memory efficient, and should be used for prefixes which do not return a huge number of objects

Returns

A list of dict describing objects found by api. The returned dict will look like this

```
[{"key": "/key/for/object",
 "size": <size_of_object_in_bytes>,
 "last_modified": <last_modified_timestamp_in_cloud_storage>,
 "metadata": Union(<metadata_dict_of_key>, None)}, ...]
```

Return type list[dict]

upload_file (*destination_key*, *source_file*, *metadata=None*)
Upload a file from local filesystem

Parameters

- **destination_key** (*str*) – Key where to store object
- **source_file** (*str*) – Path on local file system for file to be uploaded
- **metadata** (*dict*) – Metadata to be stored along with object

Returns Nothing

Return type None

upload_file_obj (*destination_key*, *source_fd*, *metadata=None*)
Upload a file from file object

Parameters

- **destination_key** (*str*) – Key where to store object
- **source_fd** (*file*) – A file object to be uploaded
- **metadata** (*dict*) – Metadata to be stored along with object

Returns Nothing

Return type None

4.1.5 Retriable Storage

Cloud storage api calls fail randomly due to various issues like timeouts. Handling retries is a common task. In spongeblob, both S3 and WABS classes implement a *get_retriable_excpetions* method which takes the method name as parameter and returns the exceptions which should be generally retried for it. *spongeblob.retriable_storage.RetriableStorage* wraps the S3/WABS client above and retries for acceptable exceptions for each method via *tenacity* library. This is just one approach to retry exceptions, and you can use it if it fits your requirement. All storage class api work directly with retriable storage as well, once the client is initialized.

class spongeblob.retriable_storage.**RetriableStorage**(*provider*, *max_attempts=3*, *wait_multiplier=2*, *max_wait_seconds=30*, **args*, ***kwargs*)

This class wraps the spongeblob storage client with tenacity library for retries.

__init__(*provider*, *max_attempts=3*, *wait_multiplier=2*, *max_wait_seconds=30*, **args*, ***kwargs*)

Intitialize a storage service which retries for *max_attempts* with exponential backoff after each attempt. After each attempt, backoff time equals 2 ^ *attempt_number* * *wait_multiplier*. This value will increase upto *max_wait_seconds*. Read tenacity docs for *wait_exponential* function for more details

Parameters

- **provider** (*str*) – Any provider supported by spongeblob
- **max_attempts** (*int*) – Maximum retry attempts
- **wait_multiplier** (*int*) – Multiplier factor for wait_exponential backoff function
- **max_wait_seconds** (*int*) – Max wait time between attempts

Example

```
from spongeblob.retriable_storage import RetriableStorage
s3 = RetriableStorage('s3',
                      aws_key='access_key_id',
                      aws_secret='access_key_secret',
                      bucket_name='testbucket')

wabs = RetriableStorage('wabs',
                        account_name='testaccount',
                        container_name='testcontainer',
                        sas_token='testtoken')
```

Index

Symbols

__init__() (spongeblob.retriable_storage.RetriableStorage method), 11
__init__() (spongeblob.storage.s3.S3 method), 8
__init__() (spongeblob.storage.wabs.WABS method), 8

C

copy_from_key() (spongeblob.storage.storage.Storage method), 8

D

delete_key() (spongeblob.storage.storage.Storage method), 9
download_file() (spongeblob.storage.storage.Storage method), 9

G

get_object_properties() (spongeblob.storage.storage.Storage method), 9

L

list_object_keys() (spongeblob.storage.storage.Storage method), 9
list_object_keys_flat() (spongeblob.storage.storage.Storage method), 10

R

RetriableStorage (class in spongeblob.retriable_storage), 10

S

S3 (class in spongeblob.storage.s3), 8
setup_storage() (in module spongeblob), 7
Storage (class in spongeblob.storage.storage), 8

U

upload_file() (spongeblob.storage.storage.Storage method), 10
upload_file_obj() (spongeblob.storage.storage.Storage method), 10

W

WABS (class in spongeblob.storage.wabs), 8