gs-wrap Documentation

Release 1.0.5

Selim Naji, Adam Radomski and Marko Ristin

May 21, 2019

Contents:

CHAPTER 1

gswrap

Wrap Google Cloud Storage API for multi-threaded data manipulation.

class gswrap.**Client**(*project=None*)

Google Cloud Storage Client for simple usage of gsutil commands.

cp(*src*, *dst*, *recursive=False*, *no_clobber=False*, *multithreaded=False*, *preserve_posix=False*)

Copy objects from source to destination URL.

Parameters

- **src** (Union[str, Path]) – Source URL
- **dst** (Union[str, Path]) – Destination URL
- **recursive** (bool) – (from https://cloud.google.com/storage/docs/gsutil/commands/cp) Causes directories, buckets, and bucket subdirectories to be copied recursively. If you neglect to use this option for an upload/download, gswrap will raise an exception and inform you that no URL matched. Same behaviour as gsutil as long as no wildcards are used.

  your-bucket before:
  ”empty”

  client.cp(src=”gs://your-bucket/some-dir/”,
  dst=”gs://your-bucket/another-dir/”, recursive=False)

  # google.api_core.exceptions.GoogleAPIError: No URLs matched

  current some-dir:
  # gs://your-bucket/some-dir/file1
  # gs://your-bucket/some-dir/dir1/file11

destination URL without slash
client.cp(src="gs://your-bucket/some-dir/",
dst="gs://your-bucket/another-dir", recursive=True)

another-dir after:
gs://your-bucket/another-dir/file1
gs://your-bucket/another-dir/dir1/file11

destination URL with slash
client.cp(src="gs://your-bucket/some-dir/",
dst="gs://your-bucket/another-dir/", recursive=True)

another-dir after:
gs://your-bucket/another-dir/some-dir/file1
gs://your-bucket/another-dir/some-dir/dir1/file11

- **no_clobber** (bool) – (from https://cloud.google.com/storage/docs/gsutil/commands/cp) When specified, existing files or objects at the destination will not be overwritten.
- **multithreaded** (bool) – if set to False the copy will be performed single-threaded. If set to True it will use multiple threads to perform the copy.
- **preserve_posix** (bool) – (from https://cloud.google.com/storage/docs/gsutil/commands/cp) Causes POSIX attributes to be preserved when objects are copied. With this feature enabled, gsutil cp will copy fields provided by stat. These are the user ID of the owner, the group ID of the owning group, the mode (permissions) of the file, and the access/modification time of the file. POSIX attributes are always preserved when blob is copied on Google Cloud Storage.

Return type None

Requires

- not contains_wildcard(prefix=str(dst))
- not contains_wildcard(prefix=str(src))

cp_many_to_many(*srcs_dsts*, *recursive=False*, *no_clobber=False*, *multithreaded=False*, *preserve_posix=False*)

Perform multiple copy operations in a single function call.

Each source will be copied to the corresponding destination. Only one function call minimizes the overhead and the operations can be performed significantly faster.

sources_destinations = [
    # Copy on Google Cloud Storage
    ('gs://your-bucket/your-dir/file',
        'gs://your-bucket/backup-dir/file'),

Copy from gcs to local

('gs://your-bucket/your-dir/file',
pathlib.Path('/home/user/storage/backup-file')),

Copy from local to gcs
(pathlib.Path('/home/user/storage/new-file'),
'gs://your-bucket/your-dir/new-file'),

Copy locally
(pathlib.Path('/home/user/storage/file'),
pathlib.Path('/home/user/storage/new-file'))]

client.cp_many_to_many(srcs_dsts=sources_destinations)

Parameters

- **srcs_dsts** (Sequence[Tuple[Union[str, Path], Union[str, Path]]]) – source URLs/paths and destination URLs/paths
- **recursive** (bool) – (from https://cloud.google.com/storage/docs/gsutil/commands/cp) Causes directories, buckets, and bucket subdirectories to be copied recursively. If you neglect to use this option for an upload/download, gswrap will raise an exception and inform you that no URL matched. Same behaviour as gsutil as long as no wildcards are used.
- **no_clobber** (bool) – (from https://cloud.google.com/storage/docs/gsutil/commands/cp) When specified, existing files or objects at the destination will not be overwritten.
- **multithreaded** (bool) – if set to False the copy will be performed single-threaded. If set to True it will use multiple threads to perform the copy.
- **preserve_posix** (bool) – (from https://cloud.google.com/storage/docs/gsutil/commands/cp) Causes POSIX attributes to be preserved when objects are copied. With this feature enabled, gsutil cp will copy fields provided by stat. These are the user ID of the owner, the group ID of the owning group, the mode (permissions) of the file, and the access/modification time of the file. POSIX attributes are always preserved when blob is copied on Google Cloud Storage.

Return type None

long_ls(*url*, *recursive=False*)
List URLs with their stats given the url.

client.long_ls(gcs_url="gs://your-bucket/your-dir", recursive=False)
# ('gs://your-bucket/your-dir/your-subdir1/', None)
# ('gs://your-bucket/your-dir/your-subdir2/' None)
# ('gs://your-bucket/your-dir/file1,
<gswrap.Stat object at 0x7fea01c4a550>)

Parameters

- **url** (str) – Google Cloud Storage URL

- **recursive** (bool) – if True, list directories recursively if False, list only direct subdirectory

Return type List[Tuple[str, Optional[*Stat*]]]

Returns List of the urls of the blobs found and their stats

Requires

- not contains_wildcard(prefix=url)
- url.startswith('gs://')

ls(*url*, *recursive=False*)

List the files on Google Cloud Storage given the prefix.

Functionality is the same as "gsutil ls (-r)" command. Except that no wildcards are allowed. For more information about "gsutil ls" check out: https://cloud.google.com/storage/docs/gsutil/commands/ls

client.ls(gcs_url="gs://your-bucket/your-dir", recursive=False)
# gs://your-bucket/your-dir/your-subdir1/
# gs://your-bucket/your-dir/your-subdir2/
# gs://your-bucket/your-dir/file1

client.ls(gcs_url="gs://your-bucket/your-dir", recursive=True)
# gs://your-bucket/your-dir/your-subdir1/file1
# gs://your-bucket/your-dir/your-subdir1/file2
# gs://your-bucket/your-dir/your-subdir2/file1
# gs://your-bucket/your-dir/file1

client.ls(url="gs://your-bucket/your-", recursive=True)
will return an empty list

Parameters

- **url** (str) – Google Cloud Storage URL
- **recursive** (bool) – List only direct subdirectories

Return type List[str]

Returns List of Google Cloud Storage URLs according the given URL

Requires

- not contains_wildcard(prefix=url)
- url.startswith('gs://')

md5_hexdigests(*urls*, *multithreaded=False*)

Retrieve hex digests of MD5 checksums for multiple URLs.

urls = ['gs://your-bucket/file1', 'gs://your-bucket/file2']
client.md5_hexdigests(urls=urls, multithreaded=False)

Parameters

- **urls** (List[str]) – URLs to stat and retrieve MD5 of
- **multithreaded** (bool) – if set to False the retrieving hex digests of md5 checksums will be performed single-threaded. If set to True it will use multiple threads to perform the this.

Return type List[Optional[str]]

Returns list of hexdigests; if an URL does not exist, the corresponding item is None.

read_bytes(*url*)

Retrieve the bytes of the blob at the URL.

The caller is expected to make sure that the file fits in memory.

data = client.read_bytes(url="gs://your-bucket/data")
data.decode('utf-8')
# I'm important data

Parameters **url** (str) – to the blob on the storage

Return type bytes

Returns bytes of the blob

Requires

- not contains_wildcard(prefix=url)
- url.startswith('gs://')

read_text(*url*, *encoding='utf-8'*)

Retrieve the text of the blob at the URL.

The caller is expected to make sure that the file fits in memory.

client.read_text(url="gs://your-bucket/file",
encoding='utf-8')
# Hello I'm text

Parameters

- **url** (str) – to the blob on the storage
- **encoding** (str) – used to decode the text, defaults to 'utf-8'

Return type str

Returns text of the blob

Requires

- not contains_wildcard(prefix=url)
- url.startswith('gs://')

rm(*url*, *recursive=False*, *multithreaded=False*)
Remove blobs at given URL from Google Cloud Storage.

# your-bucket before:
# gs://your-bucket/file
client.rm(url="gs://your-bucket/file")
# your-bucket after:
# "empty"

# your-bucket before:
# gs://your-bucket/file1
# gs://your-bucket/your-dir/file2
# gs://your-bucket/your-dir/sub-dir/file3
client.rm(url="gs://your-bucket/your-dir", recursive=True)
# your-bucket after:
# gs://your-bucket/file1

Parameters

- **url** (`str`) – Google Cloud Storage URL
- **recursive** (`bool`) – if True remove files within folders
- **multithreaded** (`bool`) – if set to False the remove will be performed single-threaded. If set to True it will use multiple threads to perform the remove.

Return type `None`

Requires

- `not contains_wildcard(prefix=url)`
- `url.startswith('gs://')`

same_md5(*path*, *url*)
Check if the MD5 differs between the local file and the blob.

client.same_md5(path='/home/user/storage/file',
url='gs://your-bucket/file')

Parameters

- **path** (`Union[str, Path]`) – to the local file
- **url** (`str`) – to the remote object in Google storage

Return type `bool`

Returns True if the MD5 is the same. False if the checksum differs or local file and/or the remote object do not exist.

Requires

- `not contains_wildcard(prefix=url)`

- `url.startswith('gs://')`

same_modtime (*path*, *url*)
Check if local path and URL have equal modification times (up to secs).

Mind that you need to copy the object with -P (preserve posix) flag.

client.same_modtime(path='/home/user/storage/file',
url='gs://your-bucket/file')

Parameters

- **path** (`Union[str, Path]`) – to the local file
- **url** (`str`) – URL to an object

Return type `bool`

Returns True if the modification time is the same

Requires

- `not contains_wildcard(prefix=url)`
- `url.startswith('gs://')`

stat (*url*)
Retrieve the stat of the object in the Google Cloud Storage.

stats = client.stat(url="gs://your-bucket/file")
stats.creation_time # 2018-11-21 13:27:46.255000+00:00
stats.update_time # 2018-11-21 13:27:46.255000+00:00
stats.content_length # 1024 [bytes]
stats.storage_class # REGIONAL
stats.file_atime # 2018-11-21 13:27:46
stats.file_mtime # 2018-11-21 13:27:46
stats.posix_uid # 1000
stats.posix_gid # 1000
stats.posix_mode # 777
stats.md5 # b'1B2M2Y8AsgTpgAmY7PhCfg=='
stats.crc32c # b'AAAAAA=='

Parameters **url** (`str`) – to the object

Return type `Optional[Stat]`

Returns object status, or None if the object does not exist or is a directory.

Requires

- `not contains_wildcard(prefix=url)`
- `url.startswith('gs://')`

write_bytes (*url*, *data*)
Write bytes to the storage at the given URL.

client.write_bytes(url="gs://your-bucket/data",
data="I'm important data".encode('utf-8'))

Parameters

- **url** (str) – where to write in the storage
- **data** (bytes) – what to write

Return type None

Returns

Requires

- not contains_wildcard(prefix=url)
- url.startswith('gs://')

write_text(*url*, *text*, *encoding='utf-8'*)

Write bytes to the storage at the given URL.

client.write_text(url="gs://your-bucket/file",
text="Hello, I'm text",
encoding='utf-8')

Parameters

- **url** (str) – where to write in the storage
- **text** (str) – what to write
- **encoding** (str) – how to encode, defaults to 'utf-8'

Return type None

Returns

Requires

- not contains_wildcard(prefix=url)
- url.startswith('gs://')

class gswrap.**Stat**

Represent stat of an object in Google Storage.

Times are given in UTC.

Variables

- **creation_time** (*Optional[datetime.datetime]*) – time when blob on Google Cloud Storage was created. Not equal creation time of the local file.
- **update_time** (*Optional[datetime.datetime]*) – time when blob on Google Cloud Storage was last updated. Not equal modification time of the local file.
- **storage_class** (*Optional[str]*) – tells in what kind of storage data is stored. More information: https://cloud.google.com/storage/docs/storage-classes
- **content_length** (*Optional[int]*) – size of the object

- **file_mtime** (*Optional[datetime.datetime]*) – modification time of the local file stored in the metadata of the blob. Only available when file was uploaded with preserve_posix.
- **file_atime** (*Optional[datetime.datetime]*) – last access time of the local file stored in the metadata of the blob. Only available when file was uploaded with preserve_posix.
- **posix_uid** (*Optional[str]*) – user id of the owner of the local file stored in the metadata of the blob. Only available when file was uploaded with preserve_posix.
- **posix_gid** (*Optional[str]*) – group id of the owner of the local file stored in the metadata of the blob. Only available when file was uploaded with preserve_posix.
- **posix_mode** (*Optional[str]*) – inode protection mode of the local file stored in the metadata of the blob. Only available when file was uploaded with preserve_posix.
- **crc32c** (*Optional[bytes]*) – CRC32C checksum for this object.
- **md5** (*Optional[bytes]*) – MD5 hash for this object.

gswrap.**contains_wildcard**(*prefix*)

Check if prefix contains any wildcards.

```
>>> contains_wildcard(prefix='gs://your-bucket/some-dir/file')
False
>>> contains_wildcard(prefix='gs://your-bucket/*/file')
True
```

Parameters **prefix** (str) – path to a file or a directory

Return type bool

Returns

gswrap.**resource_type**(*res_loc*)

Determine resource type.

```
>>> url = resource_type(res_loc='gs://your-bucket/some-dir/file')
>>> isinstance(url, _GCSURL)
True
>>> url.bucket
'your-bucket'
>>> url.prefix
'some-dir/file'
```

```
>>> path = resource_type(res_loc='/home/user/work/file')
>>> path
'/home/user/work/file'
>>> isinstance(path, str)
True
```

Parameters **res_loc** (str) – resource location

Return type Union[_GCSURL, str]

Returns class corresponding to the file/directory location

CHAPTER 2

README

2.1 gs-wrap

`gs-wrap` wraps Google Cloud Storage API for multi-threaded data manipulation including copying, reading, writing and hashing.

Originally, we used our gsutilwrap, a thin wrapper around `gsutil` command-line interface, to simplify the deployment and backup tasks related to Google Cloud Storage. However, `gsutilwrap` was prohibitively slow at copying many objects into different destinations.

Therefore we developed `gs-wrap` to accelerate these operations while keeping it equally fast or faster than `gsutilwrap` at other operations.

While the google-cloud-storage library provided by Google offers sophisticated features and good performance, its use cases and behavior differ from `gsutil`. Since we wanted the simplicity and usage patterns of `gsutil`, we created `gs-wrap`, which wraps `google-cloud-storage` in its core and with its interface set to behave like `gsutil`.

`gs-wrap` is not the first Python library wrapping Google Cloud Storage API. cloud-storage-client takes a similar approach and aims to manage both Amazon's S3 and Google Cloud Storage. Parts of it are also based on `google-cloud-storage`, however the library's behaviour differs from `gsutil` which made it hard to use as an in-place replacement for `gsutilwrap`. Additionally, the library did not offer all needed operations, for example copying to many destinations, reading, writing and hashing.

The main strength of `gs-wrap` is the ability to copy many objects from many different paths to multiple destinations, while still mimicking `gsutil` interface. A direct comparison of performance between `gs-wrap` and `gsutilwrap` can be found in the section Benchmarks.

2.2 Usage

You need to create a Google Cloud Storage bucket to use this client library. Follow along with the official Google Cloud Storage documentation to learn how to create a bucket.

2.2.1 Connect to your Google Cloud Storage bucket

First a client for interacting with the Google Cloud Storage API needs to be created. This one uses internally the Storage Client from `google-cloud-storage`.

One parameter can be passed to the client:

The Google Cloud Storage **project** which the client acts on behalf of. It will be passed when creating the internal client. If not passed, falls back to the default inferred from the locally authenticated Google Cloud SDK environment. Each project needs a separate client. Operations between two different projects are not supported.

```
import gswrap

client = gswrap.Client() # project is optional
```

2.2.2 List objects in your bucket

Warning: Wildcards (*, **, ?, [chars], [char range]) are not supported by Google Cloud Storage API and neither by `gs-wrap` at the moment [2019-01-16]. Reasons are that the `gsutil` with wildcards can hardly be equivalently reconstructed and that the toplevel search is extremely inefficient. More information about `gsutil` wildcards can be found here: https://cloud.google.com/storage/docs/gsutil/addlhelp/WildcardNames

```
client.ls(gcs_url="gs://your-bucket/your-dir", recursive=False)
# gs://your-bucket/your-dir/your-subdir1/
# gs://your-bucket/your-dir/your-subdir2/
# gs://your-bucket/your-dir/file1

client.ls(gcs_url="gs://your-bucket/your-dir", recursive=True)
# gs://your-bucket/your-dir/your-subdir1/file1
# gs://your-bucket/your-dir/your-subdir1/file2
# gs://your-bucket/your-dir/your-subdir2/file1
# gs://your-bucket/your-dir/file1
```

2.2.3 Copy objects within Google Cloud Storage

If both the source and destination URL are cloud URLs from the same provider, `gsutil` copies data "in the cloud" (i.e. without downloading to and uploading from the machine where you run `gs-wrap`).

Note: client.cp() runs single-threaded by default. When multi-threading is activated, the maximum number of workers is the number of processors on the machine, multiplied by 5. This is the multi-threading default of the ThreadPoolExecuter from the concurrent.futures library.

2.2.4 Copy file within Google Cloud Storage

```
# your-bucket before:
# gs://your-bucket/file1
client.cp(src="gs://your-bucket/file1",
          dst="gs://your-bucket/your-dir/",
          recursive=True)
# your-bucket after:
# gs://your-bucket/file1
# gs://your-bucket/your-dir/file1

# your-backup-bucket before:
# "empty"
client.cp(src="gs://your-bucket/file1",
          dst="gs://your-backup-bucket/backup-file1",
          recursive=False)
# your-backup-bucket after:
# gs://your-backup-bucket/backup-file1
```

2.2.5 Copy directory within Google Cloud Storage

```
# your-bucket before:
# "empty"
client.cp(src="gs://your-bucket/some-dir/",
dst="gs://your-bucket/another-dir/", recursive=False)
# google.api_core.exceptions.GoogleAPIError: No URLs matched

# your-bucket before:
# gs://your-bucket/some-dir/file1
# gs://your-bucket/some-dir/dir1/file11

# Destination URL without slash
client.cp(src="gs://your-bucket/some-dir/",
dst="gs://your-bucket/another-dir", recursive=True)
# your-bucket after:
# gs://your-bucket/another-dir/file1
# gs://your-bucket/another-dir/dir1/file11

# Destination URL with slash
client.cp(src="gs://your-bucket/some-dir/",
dst="gs://your-bucket/another-dir/", recursive=True)
# your-bucket after:
# gs://your-bucket/another-dir/some-dir/file1
# gs://your-bucket/another-dir/some-dir/dir1/file11

# Choose to copy multi-threaded. (default=False)
client.cp(src="gs://your-bucket/some-dir/",
dst="gs://your-bucket/another-dir", recursive=True, multithreaded=True)
# your-bucket after:
# gs://your-bucket/another-dir/file1
# gs://your-bucket/another-dir/dir1/file11
```

2.2.6 Upload objects to Google Cloud Storage

Note: **recursive** causes directories, buckets, and bucket subdirectories to be copied recursively. If you upload from local disk to Google Cloud Storage and set recursive to `False`, `gs-wrap` will raise an exception and inform you that no URL matched. This mimicks the behaviour of `gsutil` when no wildcards are used.

```
# Your local directory:
# /home/user/storage/file1
# /home/user/storage/file2
# your-bucket before:
# "empty"

client.cp(src="/home/user/storage/",
          dst="gs://your-bucket/local/",
          recursive=True)
# your-bucket after:
# gs://your-bucket/local/storage/file1
# gs://your-bucket/local/storage/file2
```

2.2.7 Download objects from Google Cloud Storage

Note: **recursive** causes directories, buckets, and bucket subdirectories to be copied recursively. If you upload from local disk to Google Cloud Storage and set recursive to `False`, `gs-wrap` will raise an exception and inform you that no URL matched. This mimicks the behaviour of `gsutil` when no wildcards are used.

```
import os
# Current your-bucket:
# gs://your-bucket/file1

client.cp(
    src="gs://your-bucket/file1",
    dst="/home/user/storage/file1")

# Your local directory:
# /home/user/storage/file1
```

2.2.8 Copy, download and upload with parameters

Note: All parameters can be used for any kind of `cp` operation.

```
# Parameter: no_clobber example:
import os

# File content before: "hello"
os.stat("/home/user/storage/file1").st_mtime # 1537947563

client.cp(
    src="gs://your-bucket/file1",
    dst="/home/user/storage/file1",
    no_clobber=True)
```

(continues on next page)

(continued from previous page)

```
# no_clobber option stops from overwriting.
# File content after: "hello"
os.stat("/home/user/storage/file1").st_mtime # 1537947563

client.cp(
    src="gs://your-bucket/file1",
    dst="/home/user/storage/file1",
    no_clobber=False)

# File content after: "hello world"
os.stat("/home/user/storage/file1").st_mtime # 1540889799

# Parameter: recursive and multi-threaded example:
# Your local directory:
# /home/user/storage/file1
# ...
# /home/user/storage/file1000
# your-bucket before:
# "empty"

# Execute normal recursive copy in multiple threads.
client.cp(src="/home/user/storage/",
          dst="gs://your-bucket/local/",
          recursive=True, multithreaded=True)
# your-bucket after:
# gs://your-bucket/local/storage/file1
# ...
# gs://your-bucket/local/storage/file1000

# Parameter: preserve_posix example:
# Your file before:
# /home/user/storage/file1
# e.g. file_mtime: 1547653413 equivalent to 2019-01-16 16:43:33

client.cp(src="/home/user/storage/file1",
          dst="gs://your-backup-bucket/file1",
          preserve_posix=False)
# your-backup-bucket after:
# gs://your-backup-bucket/file1 e.g. "no metadata file_mtime"

# Preserve the POSIX attributes. POSIX attributes are the metadata of a file.
client.cp(src="/home/user/storage/file1",
          dst="gs://your-backup-bucket/file1",
          preserve_posix=True)
# your-backup-bucket after:
# gs://your-backup-bucket/file1 e.g. file_mtime: 2019-01-16 16:43:33
```

2.2.9 Perform multiple copy operations in one call

```
sources_destinations = [
    # Copy on Google Cloud Storage
    ('gs://your-bucket/your-dir/file',
     'gs://your-bucket/backup-dir/file'),
```

(continues on next page)

(continued from previous page)

```
    # Copy from gcs to local
    ('gs://your-bucket/your-dir/file',
     pathlib.Path('/home/user/storage/backup-file')),

    # Copy from local to gcs
    (pathlib.Path('/home/user/storage/new-file'),
     'gs://your-bucket/your-dir/new-file'),

    # Copy locally
    (pathlib.Path('/home/user/storage/file'),
     pathlib.Path('/home/user/storage/new-file'))]

client.cp_many_to_many(srcs_dsts=sources_destinations)
```

2.2.10 Remove files from Google Cloud Storage

```
# your-bucket before:
# gs://your-bucket/file
client.rm(url="gs://your-bucket/file")
# your-bucket after:
# "empty"

# your-bucket before:
# gs://your-bucket/file1
# gs://your-bucket/your-dir/file2
# gs://your-bucket/your-dir/sub-dir/file3
client.rm(url="gs://your-bucket/your-dir", recursive=True)
# your-bucket after:
# gs://your-bucket/file1
```

2.2.11 Read and write files in Google Cloud Storage

```
client.write_text(url="gs://your-bucket/file",
                  text="Hello, I'm text",
                  encoding='utf-8')

client.read_text(url="gs://your-bucket/file",
                 encoding='utf-8')
# Hello I'm text

client.write_bytes(url="gs://your-bucket/data",
                   data="I'm important data".encode('utf-8'))

data = client.read_bytes(url="gs://your-bucket/data")
data.decode('utf-8')
# I'm important data
```

2.2.12 Copy os.stat() of a file or metadata of a blob

Note: POSIX attributes include meta information about a file. When copying a file locally or copying a file within Google Cloud Storage, the POSIX attributes are always preserved. On the other hand, when downloading or uploading

file to Google Cloud Storage, the POSIX attributes is only preserved when **preserve_posix** is set to True.

```
file = pathlib.Path('/home/user/storage/file')
file.touch()
print(file.stat())
# os.stat_result(st_mode=33204, st_ino=19022665, st_dev=64769, st_nlink=1,
# st_uid=1000, st_gid=1000, st_size=0, st_atime=1544015997,
# st_mtime=1544015997, st_ctime=1544015997)

# Upload does not preserve POSIX attributes.
client.cp(src=pathlib.Path('/home/user/storage/file'),
          dst="gs://your-bucket/file")

stats = client.stat(url="gs://your-bucket/file")
stats.creation_time  # 2018-11-21 13:27:46.255000+00:00
stats.update_time  # 2018-11-21 13:27:46.255000+00:00
stats.content_length  # 1024 [bytes]
stats.storage_class  # REGIONAL
stats.file_atime  # None
stats.file_mtime  # None
stats.posix_uid  # None
stats.posix_gid  # None
stats.posix_mode  # None
stats.md5  # b'1B2M2Y8AsgTpgAmY7PhCfg=='
stats.crc32c  # b'AAAAAA=='

# Upload with preserve_posix also copy POSIX attributes to blob.
# POSIX attributes are the metadata of a file.
# It also works for downloading.

client.cp(src=pathlib.Path('/home/user/storage/file'),
            dst="gs://your-bucket/file", preserve_posix=True)

stats = client.stat(url="gs://your-bucket/file")
stats.creation_time  # 2018-11-21 13:27:46.255000+00:00
stats.update_time  # 2018-11-21 13:27:46.255000+00:00
stats.content_length  # 1024 [bytes]
stats.storage_class  # REGIONAL
stats.file_atime  # 2018-11-21 13:27:46
stats.file_mtime  # 2018-11-21 13:27:46
stats.posix_uid  # 1000
stats.posix_gid  # 1000
stats.posix_mode  # 777
stats.md5  # b'1B2M2Y8AsgTpgAmY7PhCfg=='
stats.crc32c  # b'AAAAAA=='
```

2.2.13 Check correctness of copied file

```
# Check modification time when copied with preserve_posix.
client.same_modtime(path='/home/user/storage/file',
                    url='gs://your-bucket/file')

# Check md5 hash to ensure content equality.
client.same_md5(path='/home/user/storage/file', url='gs://your-bucket/file')
```

(continues on next page)

(continued from previous page)

```
# Retrieve hex digests of MD5 checksums for multiple URLs.
urls = ['gs://your-bucket/file1', 'gs://your-bucket/file2']
client.md5_hexdigests(urls=urls, multithreaded=False)
```

2.3 Documentation

The documentation is available on readthedocs.

2.4 Setup

In order to use this library, you need to go through the following steps:

1. Select or create a Cloud Platform project.
2. Enable billing for your project.
3. Enable the Google Cloud Storage API.
4. Setup Authentication using the Google Cloud SDK.

2.5 Installation

- Install gs-wrap with pip:

```
pip3 install gs-wrap
```

2.6 Development

- Check out the repository.
- In the repository root, create the virtual environment:

```
python3 -m venv venv3
```

- Activate the virtual environment:

```
source venv3/bin/activate
```

- Install the development dependencies:

```
pip3 install -e .[dev]
```

We use tox for testing and packaging the distribution. Assuming that the virtual environment has been activated and the development dependencies have been installed, run:

```
tox
```

2.6.1 Pre-commit Checks

We provide a set of pre-commit checks that lint and check code for formatting.

Namely, we use:

- yapf to check the formatting.
- The style of the docstrings is checked with pydocstyle.
- Static type analysis is performed with mypy.
- isort to sort your imports for you.
- Various linter checks are done with pylint.
- Doctests are executed using the Python doctest module.
- pyicontract-lint lints contracts in Python code defined with icontract library.
- twine to check the README for invalid markup which prevents it from rendering correctly on PyPI.

Run the pre-commit checks locally from an activated virtual environment with development dependencies:

```
./precommit.py
```

- The pre-commit script can also automatically format the code:

```
./precommit.py  --overwrite
```

2.6.2 Benchmarks

Assuming that the virtual environment has been activated, the development dependencies have been installed and the `PYTHONPATH` has been set to the project directory, run the benchmarks with:

```
./benchmark/main.py *NAME OF YOUR GCS BUCKET*
```

Here are some of our benchmark results:

Benchmark list 10000 files:

TESTED	TIME	SPEEDUP
gswrap	3.22 s	-
gsutilwrap	3.98 s	1.24 x

Benchmark upload 10000 files:

TESTED	TIME	SPEEDUP
gswrap	45.12 s	-
gsutilwrap	34.85 s	0.77 x

Benchmark upload-many-to-many 500 files:

TESTED	TIME	SPEEDUP
gswrap	2.14 s	-
gsutilwrap	65.2 s	30.49 x

Benchmark download 10000 files:

TESTED	TIME	SPEEDUP
gswrap	43.92 s	-
gsutilwrap	43.01 s	0.98 x

Benchmark download-many-to-many 500 files:

TESTED	TIME	SPEEDUP
gswrap	5.85 s	-
gsutilwrap	62.93 s	10.76 x

Benchmark copy on remote 1000 files:

TESTED	TIME	SPEEDUP
gswrap	5.09 s	-
gsutilwrap	4.47 s	0.88 x

Benchmark copy-many-to-many-on-remote 500 files:

TESTED	TIME	SPEEDUP
gswrap	6.55 s	-
gsutilwrap	62.76 s	9.57 x

Benchmark remove 1000 files:

TESTED	TIME	SPEEDUP
gswrap	3.16 s	-
gsutilwrap	3.66 s	1.16 x

Benchmark read 100 files:

TESTED	TIME	SPEEDUP
gswrap	16.56 s	-
gsutilwrap	64.73 s	3.91 x

Benchmark write 30 files:

TESTED	TIME	SPEEDUP
gswrap	2.67 s	-
gsutilwrap	32.55 s	12.17 x

Benchmark stat 100 files:

TESTED	TIME	SPEEDUP
gswrap	6.39 s	-
gsutilwrap	48.15 s	7.53 x

All results of our benchmarks can be found here.

2.7 Versioning

We follow Semantic Versioning. The version X.Y.Z indicates:

- X is the major version (backward-incompatible),
- Y is the minor version (backward-compatible), and
- Z is the patch version (backward-compatible bug fix).

CHAPTER 3

CHANGELOG

3.1 1.0.5

- Changed argument in `google.cloud.storage.client.Client.get_bucket` since interface broke with google-cloud-storage 1.16.0

3.2 1.0.4

- `crc32c` and `md5` in `Stat` are decoded into actual bytes. Google client sends crc32s and md5 of the objects base-64 encoded which is confusing for the end users.

3.3 1.0.3

- Added mypy –strict flag in precommit.py and fixed corresponding errors
- Fixed mkdir race condition when downloading
- Updated docstrings with examples
- Re-structured multi-threading execution by adding nested functions and titles for readability
- Excluded benchmark module from distribution
- Updated dependencies and re-formatted code for new yapf version

3.4 1.0.2

- Fixed downloading a file into a directory bug and added testcase
- Fixed long_ls return type

3.5 1.0.1

- Rephrase PyPi description in meta information about gs-wrap package

3.6 1.0.0

- Initial version

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Python Module Index

g

Index