S3FS Documentation

Release 1.1.0

Will McGugan

Jan 01, 2019

Contents

S3FS is a PyFilesystem interface to Amazon S3 cloud storage.

As a PyFilesystem concrete class, S3FS allows you to work with S3 in the same as any other supported filesystem.

CHAPTER 1

Installing

S3FS may be installed from pip with the following command:

```
pip install fs-s3fs
```

This will install the most recent stable version.

Alternatively, if you want the cutting edge code, you can check out the GitHub repos at https://github.com/pyfilesystem/s3fs

CHAPTER 2

Opening an S3 Filesystem

There are two options for constructing a s3fs instance. The simplest way is with an *opener*, which is a simple URL like syntax. Here is an example:

```
from fs import open_fs
s3fs = open_fs('s3://mybucket/')
```

For more granular control, you may import the S3FS class and construct it explicitly:

```
from fs_s3fs import S3FS
s3fs = S3FS('mybucket')
```

2.1 S3FS Constructor

class fs_s3fs.**S3FS**(*bucket_name*, *dir_path=u'/'*, *aws_access_key_id=None*, *aws_secret_access_key=None*, *aws_session_token=None*, *endpoint_url=None*, *region=None*, *delimiter=u'/'*, *strict=True*, *cache_control=None*, *acl=None*, *upload_args=None*, *download_args=None*)

Construct an Amazon S3 filesystem for PyFilesystem

Parameters

- **bucket_name** (*str*) – The S3 bucket name.
- **dir_path** (*str*) – The root directory within the S3 Bucket. Defaults to `"/"`
- **aws_access_key_id** (*str*) – The access key, or `None` to read the key from standard configuration files.
- **aws_secret_access_key** (*str*) – The secret key, or `None` to read the key from standard configuration files.
- **endpoint_url** (*str*) – Alternative endpoint url (`None` to use default).
- **aws_session_token** (*str*) –

- **region** (*str*) – Optional S3 region.
- **delimiter** (*str*) – The delimiter to separate folders, defaults to a forward slash.
- **strict** (*bool*) – When `True` (default) S3FS will follow the PyFilesystem specification exactly. Set to `False` to disable validation of destination paths which may speed up uploads / downloads.
- **cache_control** (*str*) – Sets the 'Cache-Control' header for uploads.
- **acl** (*str*) – Sets the Access Control List header for uploads.
- **upload_args** (*dict*) – A dictionary for additional upload arguments. See https://boto3.readthedocs.io/en/latest/reference/services/s3.html#S3.Object.put for details.
- **download_args** (*dict*) – Dictionary of extra arguments passed to the S3 client.

CHAPTER 3

Limitations

Amazon S3 isn't strictly speaking a *filesystem*, in that it contains files, but doesn't offer true *directories*. S3FS follows the convention of simulating directories by creating an object that ends in a forward slash. For instance, if you create a file called *"foo/bar"*, S3FS will create an S3 object for the file called *"foo/bar" and* an empty object called *"foo/"* which stores that fact that the *"foo"* directory exists.

If you create all your files and directories with S3FS, then you can forget about how things are stored under the hood. Everything will work as you expect. You *may* run in to problems if your data has been uploaded without the use of S3FS. For instance, if you create a *"foo/bar"* object without a *"foo/"* object. If this occurs, then S3FS may give errors about directories not existing, where you would expect them to be. The solution is to create an empty object for all directories and subdirectories. Fortunately most tools will do this for you, and it is probably only required of you upload your files manually.

CHAPTER 4

Authentication

If you don't supply any credentials, then S3FS will use the access key and secret key configured on your system. You may also specify when creating the filesystem instance. Here's how you would do that with an opener:

```
s3fs = open_fs('s3://<access key>:<secret key>@mybucket')
```

Here's how you specify credentials with the constructor:

```
s3fs = S3FS(
    'mybucket'
    aws_access_key_id=<access key>,
    aws_secret_access_key=<secret key>
)
```

Note: Amazon recommends against specifying credentials explicitly like this in production.

CHAPTER 5

S3 Info

You can retrieve S3 info via the `s3` namespace. Here's an example:

```
>>> info = s.getinfo('foo', namespaces=['s3'])
>>> info.raw['s3']
{'metadata': {}, 'delete_marker': None, 'version_id': None, 'parts_count': None,
↪'accept_ranges': 'bytes', 'last_modified': 1501935315, 'content_length': 3,
↪'content_encoding': None, 'request_charged': None, 'replication_status': None,
↪'server_side_encryption': None, 'expires': None, 'restore': None, 'content_type':
↪'binary/octet-stream', 'sse_customer_key_md5': None, 'content_disposition': None,
↪'storage_class': None, 'expiration': None, 'missing_meta': None, 'content_language
↪': None, 'ssekms_key_id': None, 'sse_customer_algorithm': None, 'e_tag': '
↪"37b51d194a7513e45b56f6524f2d51f2"', 'website_redirect_location': None, 'cache_
↪control': None}
```

CHAPTER 6

URLs

You can use the `geturl` method to generate an externally accessible URL from an S3 object. Here's an example:

```
>>> s3fs.geturl('foo')
'https://fsexample.s3.amazonaws.com//foo?AWSAccessKeyId=AKIAIEZZDQU72WQP3JUA&
↪Expires=1501939084&Signature=4rfDuqVgmvILjtTeYOJvyIXRMvs%3D'
```

CHAPTER 7

More Information

See the PyFilesystem Docs for documentation on the rest of the PyFilesystem interface.

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

Index

S