dj-libcloud Documentation

Release 0.2.0

Daniel Greenfeld

Sep 27, 2017

Contents

1 dj-libcloud **3**
- 1.1 Documentation . . . 3
- 1.2 Quickstart . . . 3
- 1.3 Other LibCloud Providers . . . 4
- 1.4 Features . . . 4
- 1.5 FAQ . . . 4
- 1.6 CREDIT . . . 5

2 Cookbook **7**
- 2.1 Amazon . . . 7
- 2.2 Google Cloud Storage . . . 7
- 2.3 Rackspace Cloudfiles . . . 8
- 2.4 Microsoft Azure . . . 8
- 2.5 Using django-pipeline . . . 8

3 Contributing **11**
- 3.1 Types of Contributions . . . 11
- 3.2 Get Started! . . . 12
- 3.3 Pull Request Guidelines . . . 13
- 3.4 Tips . . . 13

4 Credits **15**
- 4.1 Development Lead . . . 15
- 4.2 Contributors . . . 15

5 History **17**
- 5.1 0.2.0(2014-06-23) . . . 17
- 5.2 0.1.2 (2014-04-24) . . . 17
- 5.3 0.1.1 (2014-04-21) . . . 17
- 5.4 0.1.0 (2014-04-21) . . . 18

Contents:

CHAPTER 1

dj-libcloud

Adds easy python 3 and 2.7 support to Django for management of static assets. This is a wrapper around the excellent Apache Libcloud library.

Documentation

The full documentation is at https://dj-libcloud.readthedocs.org.

Quickstart

Libcloud verifies server SSL certificates before it lets you do anything. It will search your system for the CA certificate, and if it doesn't find it then it will blow up. See https://libcloud.readthedocs.org/en/latest/other/ssl-certificate-validation.html

Installing CA certificate bundle on Mac OS X:

```
# Assuming you are using homebrew for Mac OS X dependency management.
$ brew install curl-ca-bundle
```

Install dj-libcloud:

```
$ pip install dj-libcloud
```

Then use it in a project, e.g. for your static files:

```
# settings.py

STATIC_URL = 'https://s3.amazonaws.com/my-assets/'
STATICFILES_STORAGE = 'djlibcloud.storage.LibCloudStorage'

LIBCLOUD_PROVIDERS = {
    'default': {
```

```
        'type': 'libcloud.storage.types.Provider.S3',
        'user': os.environ.get('AWS_ACCESS_KEY'),
        'key': os.environ.get('AWS_SECRET_KEY'),
        'bucket': 'my-assets',
        'secure': True,
    },
}
```

Other LibCloud Providers

If you want to use other libcloud providers (Rackspace, Openstack, other AWS centers, et al), please visit:

- The libcloud list of supported providers
- The dj-libcloud cookbook.

Features

- Works for uploading media assets using Python 3.3 and Django 1.6.
- In theory supports all the backends that libcloud supports.

FAQ

Because you just had to ask.

Why not use dj-static or whitenoise?

Those are great libraries, but are not what you want when handling user uploaded media.

Why not just update django-storages?

libcloud is awesome and has a dedicated team devoted to it. We can have it do most of the heavy lifting. On the other hand, converting *django-storages* to work with Python 3 looked like too much work. Sometimes you just have to start anew, right?

What storage providers does dj-libcloud support?

dj-libcloud is a wrapper around libcloud, meaning it supports all the providers of that library. Check out the full list of supported providers!

How can I contribute?

Please read http://dj-libcloud.readthedocs.org/en/latest/contributing.html

What about compressors like django-pipeline?

Working on it. Currently the *PipelineCachedCloudStorage* class breaks the second time you run it. See https://github.com/pydanny/dj-libcloud/issues/7

CREDIT

Many thanks to Jannis Leidel (@jezdez) for giving me the code to get this started. He's a Django core developer, the master of Django static asset managment, and overall a great great guy.

CHAPTER 2

Cookbook

Amazon

Using eu-west-1

```
STATIC_URL = 'https://s3-eu-west-1.amazonaws.com/my-assets/'
STATICFILES_STORAGE = 'djlibcloud.storage.LibCloudStorage'

LIBCLOUD_PROVIDERS = {
    'amazon_s3_eu_west': {
        'type': 'libcloud.storage.types.Provider.S3_EU_WEST',
        'user': os.environ.get('AWS_ACCESS_KEY'),
        'key': os.environ.get('AWS_SECRET_KEY'),
        'bucket': 'my-assets',
        'secure': True,
    },
}

DEFAULT_LIBCLOUD_PROVIDER = 'amazon_s3_eu_west'
```

Google Cloud Storage

```
STATIC_URL = 'https://storage.googleapis.com/my-assets/'
STATICFILES_STORAGE = 'djlibcloud.storage.LibCloudStorage'

LIBCLOUD_PROVIDERS = {
    'google_cloud_storage': {
        'type': 'libcloud.storage.types.Provider.GOOGLE_STORAGE',
        'user': os.environ.get('GOOGLE_ACCESS_KEY'),
        'key': os.environ.get('GOOGLE_SECRET_KEY'),
        'bucket': 'my-assets',
```

```
        'secure': True,
    },
}

DEFAULT_LIBCLOUD_PROVIDER = 'google_cloud_storage'
```

Rackspace Cloudfiles

```
STATIC_URL = 'https://<long>-<hash>.ssl.cf5.rackcdn.com'
STATICFILES_STORAGE = 'djlibcloud.storage.LibCloudStorage'

LIBCLOUD_PROVIDERS = {
    'rackspace_cloudfiles': {
        'type': 'libcloud.storage.types.Provider.CLOUDFILES',
        'user': os.environ.get('RACKSPACE_USER_NAME'),
        'key': os.environ.get('RACKSPACE_API_KEY'),
        'bucket': 'my-assets',
        'secure': True,
    },
}

DEFAULT_LIBCLOUD_PROVIDER = 'rackspace_cloudfiles'
```

Microsoft Azure

```
AZURE_ACCOUNT_NAME = os.environ.get('AZURE_ACCOUNT_NAME')
STATIC_URL = 'https://%s.blob.core.windows.net/my-assets/' % AZURE_ACCOUNT_NAME
STATICFILES_STORAGE = 'djlibcloud.storage.LibCloudStorage'

LIBCLOUD_PROVIDERS = {
    'microsoft_azure': {
        'type': 'libcloud.storage.types.Provider.AZURE_BLOBS',
        'user': AZURE_ACCOUNT_NAME,
        'key': os.environ.get('AZURE_ACCOUNT_KEY'),
        'bucket': 'my-assets',
        'secure': True,
    },
}

DEFAULT_LIBCLOUD_PROVIDER = 'microsoft_azure'
```

Using django-pipeline

```
# core/storage.py

from djlibcloud.storage import LibCloudStorage
from pipeline.storage import PipelineMixin

class PipelineCloudStorage(PipelineMixin,
```

```
                            LibCloudStorage):
    """ UNTESTED! """
    pass
```

```
# settings.py
STATICFILES_STORAGE = 'core.storage.PipelineCloudStorage'
```

CHAPTER 3

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

Types of Contributions

Report Bugs

Report bugs at https://github.com/pydanny/dj-libcloud/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" is open to whoever wants to implement it.

Implement Features

Look through the GitHub issues for features. Anything tagged with "feature" is open to whoever wants to implement it.

Write Documentation

dj-libcloud could always use more documentation, whether as part of the official dj-libcloud docs, in docstrings, or even on the web in blog posts, articles, and such.

Submit Feedback

The best way to send feedback is to file an issue at https://github.com/pydanny/dj-libcloud/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

Get Started!

Ready to contribute? Here's how to set up *dj-libcloud* for local development.

1. Fork the *dj-libcloud* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/dj-libcloud.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv dj-libcloud
$ cd dj-libcloud/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 djlibcloud tests
$ python setup.py test
$ tox
```

To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.7, and 3.3. Check https://travis-ci.org/pydanny/dj-libcloud/pull_requests and make sure that the tests pass for all supported Python versions.

Tips

To run a subset of tests:

```
$ python -m unittest tests.test_djlibcloud
```

CHAPTER 4

Credits

Development Lead

- Daniel Greenfeld <pydanny@gmail.com>
- Jannis Leidel

Contributors

None yet. Why not be the first?

CHAPTER 5

History

0.2.0(2014-06-23)

- Updated url method to return correct values for Rackspace Cloudfiles, Microsoft Azure, Google Storage, non-US Amazon SV (Thanks Jannis!)
- Improved the cookbook immensely (Thanks Jannis!)
- Add link from README to cookbook
- Fix `url` method for Google Storage, Rackspace Cloudfiles and Microsoft Azure. Also return the correct value for non-US Amazon S3 buckets.

0.1.2 (2014-04-24)

- Confirmed to work with Python 2.7
- Remove django-pipeline specific code from storages
- Add cookbook to docs that includes django-pipeline

0.1.1 (2014-04-21)

- Fixed second-time run problem by just using LibCloudStorage class
- Made django-pipeline optional
- Removed unnecessary files
- Moved TODO to issue tracker

0.1.0 (2014-04-21)

- First release on PyPI.
- Frustration over lack of easy media asset support for Django and Python 3.