django-cumulus Documentation

Release 1.0.5

Rich Leland

December 01, 2013

Contents

The aim of django-cumulus is to provide a set of tools to utilize Rackspace Cloud Files through Django. It currently includes a custom file storage class, CloudFilesStorage.

CHAPTER 1

changelog

1.1 Version 1.0.5, 30 January 2012

- Added CloudFilesStaticStorage subclass for collectstatic compatability
- Added thread-safe CloudFilesStorage subclass
- Added four new management commands
- Added creation of pseudo-directories
- Numerous bug fixes and code cleanups
- Created new example project based on Django 1.3
- Updated tox configuration
- Bumped python-cloudfiles requirement to 1.7.9.3

1.2 Version 1.0.4, 02 September 2011

- Added USE_SSL setting, which outputs SSL URLs. Thanks to @whafro for the nudge.

1.3 Version 1.0.3, 07 June 2011

- Added context processor for using container URLs in templates for statically synced media

1.4 Version 1.0.2, 05 May 2011

- Added `CUMULUS_CNAMES` setting to map cloudfiles URIs to CNAMEs

1.5 Version 1.0, 03 March 2011

- OK, srsly. Time for 1.0.
- Fixed content_type bug
- Bumped python-cloudfiles requirement to 1.7.8

1.6 Version 0.3.6, 19 January 2011

- Added containerinfo management command
- Properly integrated `CUMULUS_USE_SERVICENET` into storage backend
- Resolved issue 5, adding `CUMULUS_TIMEOUT` setting to specify default connection timeout
- Restructured tests to work properly with django-nose

1.7 Version 0.3.5, 07 January 2011

- Fixed glaring issue affecting Django > 1.1.x (see http://bit.ly/e8YhcR)
- Removed reliance on physical files for tests
- Added tox config to test multiple versions of Python and Django

1.8 Version 0.3.4, 13 September 2010

- Reverted exception handling to pre-2.6 style
- Added example project to repo

1.9 Version 0.3.3, 12 July 2010

- Removed reliance on bitbucket tag download files

1.10 Version 0.3.2, 12 July 2010

- Pulled in Ian Schenck's delete_object fix

1.11 Version 0.3.1, 18 May 2010

- Fixed syncstatic deletion bug
- Require verbosity > 1 for syncstatic output

1.12 Version 0.3, 17 May 2010

- Added syncstatic management command

1.13 Version 0.2.3, 03 May 2010

- Fix bug when accessing imagekit attributes
- Fix setup.py distribute installation issue

1.14 Version 0.2.2, 11 February 2010

- Fixed bug when using django-imagekit

1.15 Version 0.2, 10 February 2010

- Changed focus and aim of project
- Removed all previous custom admin work
- Incorporated CloudFilesStorage custom storage backend
- Added sphinx docs
- Converted setup to use distribute

1.16 Version 0.1, 28 July 2009

- Initial release

CHAPTER 2

Installation

To install the latest release from PyPI using pip:

```
pip install django-cumulus
```

To install the development version using pip:

```
pip install -e git://github.com/richleland/django-cumulus.git#egg=django-cumulus
```

Add `cumulus` to `INSTALLED_APPS`:

```
INSTALLED_APPS = (
    ...
    'cumulus',
    ...
)
```

CHAPTER 3

Usage

Add the following to your project's settings.py file:

```
CUMULUS = {
    'USERNAME': 'YourUsername',
    'API_KEY': 'YourAPIKey',
    'CONTAINER': 'ContainerName'
}
DEFAULT_FILE_STORAGE = 'cumulus.storage.CloudFilesStorage'
```

Alternatively, if you don't want to set the DEFAULT_FILE_STORAGE, you can do the following in your models:

```
from cumulus.storage import CloudFilesStorage

cloudfiles_storage = CloudFilesStorage()

class Photo(models.Model):
    image = models.ImageField(storage=cloudfiles_storage, upload_to='photos')
    alt_text = models.CharField(max_length=255)
```

Then access your files as you normally would through templates:

```
<img src="{{ photo.image.url }}" alt="{{ photo.alt_text }}" />
```

Or through Django's default ImageField or FileField api:

```
>>> photo = Photo.objects.get(pk=1)
>>> photo.image.width
300
>>> photo.image.height
150
>>> photo.image.url
http://c0000000.cdn.cloudfiles.rackspacecloud.com/photos/some-image.jpg
```

CHAPTER 4

Static media

django-cumulus will work with Django's built-in `collectstatic` management command out of the box. You need to supply a few additional settings:

```
CUMULUS = {
    'STATIC_CONTAINER': 'YourStaticContainer'
}
STATICFILES_STORAGE = 'cumulus.storage.CloudFilesStaticStorage'
```

CHAPTER 5

Context Processor

django-cumulus includes an optional context_processor for accessing the full CDN_URL of any container files from your templates.

This is useful when you're using Cloud Files to serve you static media such as css and javascript and don't have access to the `ImageField` or `FileField`'s url() convenience method.

Add `cumulus.context_processors.cdn_url` to your list of context processors in your project's settings.py file:

```
TEMPLATE_CONTEXT_PROCESSORS = (
    ...
    'cumulus.context_processors.cdn_url',
    ...
)
```

Now in your templates you can use {{ CDN_URL }} to output the full path to local media:

```
<link rel="stylesheet" href="{{ CDN_URL }}css/style.css">
```

CHAPTER 6

Management commands

6.1 syncstatic

This management command synchronizes a local static media folder with a remote container. A few extra settings are required to make use of the command.

Add the following required settings:

```
CUMULUS = {
    'STATIC_CONTAINER': 'MyStaticContainer', # the name of the container to sync with
    'SERVICENET': False, # whether to use rackspace's internal private network
    'FILTER_LIST': [] # a list of files to exclude from sync
}
```

Invoke the management command:

```
django-admin.py syncstatic
```

You can also perform a test run:

```
django-admin.py syncstatic -t
```

For a full list of available options:

```
django-admin.py help syncstatic
```

6.2 container_create

This management command creates a new container in Cloud Files.

Invoke the management command:

```
django-admin.py container_create <container_name>
```

For a full list of available options:

```
django-admin.py help container_create
```

6.3 container_delete

This management command deletes a container in Cloud Files.

Invoke the management command:

```
django-admin.py container_delete <container_name>
```

For a full list of available options:

```
django-admin.py help container_delete
```

6.4 container_info

This management command gathers information about containers in Cloud Files.

Invoke the management command:

```
django-admin.py container_info [<container_one> <container two> ...]
```

For a full list of available options:

```
django-admin.py help container_info
```

6.5 container_list

This management command lists all the items in a Cloud Files container to stdout.

Invoke the management command:

```
django-admin.py container_list <container_name>
```

For a full list of available options:

```
django-admin.py help container_list
```

CHAPTER 7

Settings

Below are the default settings:

```
CUMULUS = {
    'API_KEY': None,
    'AUTH_URL': 'us_authurl',
    'CNAMES': None,
    'CONTAINER': None,
    'SERVICENET': False,
    'TIMEOUT': 5,
    'TTL': 86400,
    'USE_SSL': False,
    'USERNAME': None,
    'STATIC_CONTAINER': None,
    'FILTER_LIST': []
}
```

7.1 API_KEY

Required. This is your API access key. You can obtain it from the Rackspace Management Console.

7.2 AUTH_URL

Set this to the region your account is in. Valid values are `us_authurl` (default) and `uk_authurl`.

7.3 CNAMES

A mapping of ugly Rackspace URLs to CNAMEd URLs. Example:

```
CUMULUS = {
    'CNAMES': {
        'http://c3417812.r12.cf0.rackcdn.com': 'http://media.mysite.com'
    }
}
```

7.4 CONTAINER

Required. The name of the container you want files to be uploaded to.

7.5 FILTER_LIST

A list of items to exclude when using the `syncstatic` management command. Defaults to an empty list.

7.6 SERVICENET

Specifies whether to use Rackspace's private network (True) or not (False). If you host your sites on Rackspace, you should set this to True in production as you will not incur data transfer fees between your server(s) and Cloud Files on the private network.

7.7 STATIC_CONTAINER

When using Django's `collectstatic` or django-cumulus's `syncstatic` command, this is the name of the container you want static files to be uploaded to.

7.8 TIMEOUT

The timeout to use when attempting connections to Cloud Files. Defaults to 5 (seconds).

7.9 TTL

The maximum time (in seconds) until a copy of one of your files distributed into the CDN is re-fetched from your container. Defaults to 86400 (seconds) (24h), the default set by python-cloudfiles.

Note: After changing TTL, caching servers may not recognize the new TTL for this container until the previous TTL expires.

7.10 USE_SSL

Whether or not to retrieve the container URL as http (False) or https (True).

7.11 USERNAME

Required. This is your API username. You can obtain it from the Rackspace Management Console.

7.12 HEADERS

Set headers based on a regular expression in the file name. This can be used to allow Firefox to access webfonts across domains:

```
CUMULUS = {
    'HEADERS': (
        (r'.*\.(eot|otf|woff|ttf)$', {
            'Access-Control-Allow-Origin': '*'
        }),
    )
}
```

CHAPTER 8

Requirements

- Django>=1.2
- python-cloudfiles >= 1.7.9

CHAPTER 9

Tests

To run the tests, clone the github repo, install tox and invoke `tox` from the clone's root. This will upload two very small files to your container and delete them when the tests have finished running.

CHAPTER 10

Issues

To report issues, please use the issue tracker at https://github.com/richleland/django-cumulus/issues.