django-cumulus Documentation

Release 1.0.20-6-gc285619-dirty

Rich Leland

Sep 01, 2018

Contents

`django-cumulus` provides a set of tools to utilize the python-swiftclient and Rackspace Cloud Files API from Django. It includes a custom file storage class, CumulusFilesStorage.

More documentation about the usage and installation of `django-cumulus` can be found on django-cumulus.readthedocs.org.

The source code for `django-cumulus` can be found and contributed to on github.com/django-cumulus/django-cumulus. There you can also file issues.

This documentation applies to the version 1.0.20-6-gc285619-dirty of django-cumulus. To find out what's new in this version, please see the changelog

CHAPTER 1

changelog

1.1 Version 1.2, (in progress)

- Revive the changelog
- Rename legacy SwiftclientStorage to CumulusStorage (keeping backwards compatibility)
- Fix Django 1.7 support
- Bugfixes

1.2 Version 1.0.13, 1 September 2014

- Revive the changelog
- Among other things django-cumulus is based on pyrax
- New people taking care of this project now
- Too much history to list here, sorry

1.3 Version 1.0.5, 30 January 2012

- Combine syncstatic and syncmedia into to syncfiles (backwards incompatible change)
- 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

1.4 Version 1.0.4, 02 September 2011

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

1.5 Version 1.0.3, 07 June 2011

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

1.6 Version 1.0.2, 05 May 2011

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

1.7 Version 1.0, 03 March 2011

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

1.8 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

1.9 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

1.10 Version 0.3.4, 13 September 2010

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

1.11 Version 0.3.3, 12 July 2010

- Removed reliance on bitbucket tag download files

1.12 Version 0.3.2, 12 July 2010

- Pulled in Ian Schenck's delete_object fix

1.13 Version 0.3.1, 18 May 2010

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

1.14 Version 0.3, 17 May 2010

- Added syncstatic management command

1.15 Version 0.2.3, 03 May 2010

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

1.16 Version 0.2.2, 11 February 2010

- Fixed bug when using django-imagekit

1.17 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

1.18 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/django-cumulus/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',
    'PYRAX_IDENTITY_TYPE': 'rackspace',
}
DEFAULT_FILE_STORAGE = 'cumulus.storage.CumulusStorage'
```

The `PYRAX_IDENTITY_TYPE` parameter can either be `rackspace` or `keystone` depending on whether you use Rackspace or OpenStack respectively.

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

```
from cumulus.storage import CumulusStorage

swiftclient_storage = CumulusStorage()

class Photo(models.Model):
    image = models.ImageField(storage=swiftclient_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
```

(continues on next page)

(continued from previous page)

```
>>> photo.image.url
http://c0000000.cdn.cloudfiles.rackspacecloud.com/photos/some-image.jpg
```

CHAPTER 4

Static files

`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.CumulusStaticStorage'
```

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 Cumulus 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 syncfiles

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

Add the following required settings:

```
CUMULUS = {
    'CONTAINER': 'MyMediaContainer', # the name of the media container to sync with
    'STATIC_CONTAINER': 'MyStaticContainer',  # the name of the static container to␣
↪sync with
    'SERVICENET': False,  # whether to use rackspace's internal private network
}
```

Invoke the management command:

```
django-admin.py syncfiles --static

django-admin.py syncfiles --media
```

You can also perform a test run:

```
django-admin.py syncfiles --test-run
```

For a full list of available options:

```
django-admin.py help syncfiles
```

6.2 container_create

This management command creates a new container.

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.

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:

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 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',
    'AUTH_VERSION': '1.0',
    'AUTH_TENANT_NAME': None,
    'AUTH_TENANT_ID': None,
    'REGION': 'DFW',
    'CNAMES': None,
    'CONTAINER': None,
    'CONTAINER_URI': None,
    'CONTAINER_SSL_URI': None,
    'SERVICENET': False,
    'TIMEOUT': 5,
    'TTL': 86400,
    'USE_SSL': False,
    'USERNAME': None,
    'STATIC_CONTAINER': None,
    'INCLUDE_LIST': [],
    'EXCLUDE_LIST': [],
    'HEADERS': {},
    'GZIP_CONTENT_TYPES': [],
    'USE_PYRAX': True,
    'PYRAX_IDENTITY_TYPE': None,
}
```

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`, or if you are not using rackspace, your swift auth url.

7.3 AUTH_VERSION

OpenStack auth version to use with the `swiftclient`. Does not apply to `pyrax` based connections.

7.4 AUTH_TENANT_NAME and AUTH_TENANT_ID

Required if you are using your own Openstack Swift rather than rackspaces.

7.5 REGION

Set this to the regional datacenter to connect to. Valid values are `DFW` (default) `ORD` and `LON`.

7.6 CNAMES

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

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

7.7 CONTAINER

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

7.8 CONTAINER_URI and CONTAINER_SSL_URI

Specified URLs for the container will be used instead of looking up the URL directly from the container.

7.9 INCLUDE_LIST

A list of glob-style regular expresions to match files or directories to include when using the `syncfiles` management command. Defaults to an empty list.

7.10 EXCLUDE_LIST

A list of glob-style regular expresions to match files or directories to exclude when using the `syncfiles` management command. Defaults to an empty list.

7.11 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 the cdn on the private network.

7.12 STATIC_CONTAINER

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

7.13 TIMEOUT

The timeout to use when attempting connections over swiftclient. Defaults to 5 (seconds).

7.14 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 pyrax.

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

7.15 USE_SSL

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

7.16 USERNAME

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

7.17 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': '*'
        }),
    )
}
```

7.18 GZIP_CONTENT_TYPES

Set which content types must be gzipped before sent to the cloud:

```
CUMULUS = {
    'GZIP_CONTENT_TYPES': ['image/jpeg', 'text/css'],
}
```

The files matching these content types would be gzipped and will have *gzip* content-encoding.

7.19 USE_PYRAX

If True, will use the Official Rackspace's Python SDK for OpenStack/Rackspace APIs. Defaults to True.

Note: Currently this is required even to use your own OpenStack Swift setup.

7.20 PYRAX_IDENTITY_TYPE

Pyrax supports different identity types. For now (version 1.4.5 of Pyrax), there are two types available: *rackspace* and *keystone*.

You **can** specify it through cumulus settings and if you don't, you **must** do it through other means (like environment variables or configuration files, see Pyrax documentation for more details).

CHAPTER 8

Requirements

- Django>=1.4
- pyrax>=1.9,<1.10

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

The source code for `django-cumulus` can be found and contributed to on github.com/django-cumulus/django-cumulus. There you can also file issues.

CHAPTER 11

Changelog

To find out what's new in this version of django-cumulus, please see the changelog