djcloudbridge Documentation

Release 0.3.0

Galaxy
GVL Projects

Dec 25, 2017

Contents

Contents:

CHAPTER 1

djcloudbridge

A reusable Django app that exposes a ReSTful Web API for interacting with CloudBridge providers. The structure of the API mirrors the organisation of CloudBridge's API and allows for creating, retrieving and updating CloudBridge resources.

1.1 Documentation

The full documentation is at https://djcloudbridge.readthedocs.io.

1.2 Quickstart

Install djcloudbridge:

```
pip install djcloudbridge
```

Add it to your *INSTALLED_APPS*:

```
INSTALLED_APPS = (
    ...
    'djcloudbridge.apps.DjangoCloudbridgeConfig',
    ...
)
```

Add djcloudbridge's URL patterns:

```
from djcloudbridge import urls as djcloudbridge_urls


urlpatterns = [
    ...
    url(r'^', include(djcloudbridge_urls)),
```

```
    ...
]
```

And finally, the following settings are recommended in your settings.py

```
REST_FRAMEWORK = {
    'PAGE_SIZE': 50,
    'DEFAULT_PAGINATION_CLASS': 'rest_framework.pagination.PageNumberPagination',
    'DEFAULT_AUTHENTICATION_CLASSES': (
        'rest_framework.authentication.SessionAuthentication',
        'rest_framework.authentication.TokenAuthentication'
    )
}

REST_AUTH_SERIALIZERS = {
    'USER_DETAILS_SERIALIZER': 'djcloudbridge.serializers.UserSerializer'
}

REST_SESSION_LOGIN = True

# **Make sure to change** the value for ``FERNET_KEYS`` variable
# because it is used to encrypt sensitive database fields.
FERNET_KEYS = [
    'new key for encrypting'
]
```

1.3 Running the API Locally

You can run a test server to browse the API endpoints locally. DJCloudBridge is based on Python 3.6 and although it may work on older Python versions, 3.6 is the only supported version. Use of virtualenv is also highly advised.

To get started, simply register the provider connection information under the relevant cloud model (e.g. AWS, Azure, GCE, OpenStack) in Django Admin. Then create a User Profile under the User Profile model. Finally, use the API browser at http://localhost:8000/clouds to view the cloud you registered and interact with cloud resources for that provider.

1. Checkout djcloudbridge and create environment

```
$ mkdir djcloudbridge && cd djcloudbridge
$ virtualenv -p python3.6 venv --prompt "(djcloudbridge)" && source venv/bin/activate
$ git clone https://github.com/cloudve/djcloudbridge.git
$ cd djcloudbridge
$ pip install -r requirements.txt
$ python manage.py migrate
$ python manage.py createsuperuser
$ python manage.py runserver
```

2. Visit http://127.0.0.1:8000/admin/ to define your cloud connection settings.
3. Visit http://127.0.0.1:8000/clouds/ to explore the API.

1.4 Features

- TODO

1.5 Running Tests

Does the code actually work?

```
source <YOURVIRTUALENV>/bin/activate
(myenv) $ pip install tox
(myenv) $ tox
```

1.6 Credits

Tools used in rendering this package:

- Cookiecutter
- cookiecutter-djangopackage

CHAPTER 2

Installation

At the command line:

```
$ easy_install djcloudbridge
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv djcloudbridge
$ pip install djcloudbridge
```

CHAPTER 3

Usage

To use djcloudbridge in a project, add it to your *INSTALLED_APPS*:

```
INSTALLED_APPS = (
    ...
    'djcloudbridge.apps.DjangoCloudbridgeConfig',
    ...
)
```

Add djcloudbridge's URL patterns:

```
from djcloudbridge import urls as djcloudbridge_urls


urlpatterns = [
    ...
    url(r'^', include(djcloudbridge_urls)),
    ...
]
```

CHAPTER 4

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:

4.1 Types of Contributions

4.1.1 Report Bugs

Report bugs at https://github.com/cloudve/djcloudbridge/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.

4.1.2 Fix Bugs

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

4.1.3 Implement Features

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

4.1.4 Write Documentation

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

4.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/cloudve/djcloudbridge/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 :)

4.2 Get Started!

Ready to contribute? Here's how to set up *djcloudbridge* for local development.

1. Fork the *djcloudbridge* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/djcloudbridge.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 djcloudbridge
$ cd djcloudbridge/
$ 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 djcloudbridge 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.

4.3 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 Python3.6, and for PyPy. Check https://travis-ci.org/cloudve/djcloudbridge/pull_requests and make sure that the tests pass for all supported Python versions.

4.4 Tips

To run a subset of tests:

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

CHAPTER 5

History

5.1 0.1.0 (2017-10-04)

- First release on PyPI.

5.2 0.2.0 (2017-11-15)

- Updated AWS cloud model to reflect cloudbridge changes.
- Minor bug fixes

5.3 0.3.0 (2017-12-17)

- Moved azure resource group, storage account and vm_default_username to credentials
- Changed status to beta