
Cumulus Documentation

Release 0.1.18

Brett Swift

Mar 12, 2019

Contents:

1	Cumulus	1
1.1	Features	1
1.2	Credits	1
2	Installation	3
2.1	Stable release	3
2.2	From sources	3
3	Usage	5
4	Contributing	7
4.1	Submit Feedback	7
5	Credits	11
5.1	Development Lead	11
5.2	Contributors	11
6	Indices and tables	13

CHAPTER 1

Cumulus

Simplified dynamic cloudformation resources with an opinion

- Free software: MIT license
- Documentation: https://cfn_cumulus.readthedocs.io.

1.1 Features

- TODO

1.2 Credits

This package was created with [Cookiecutter](#) and the [audreyr/cookiecutter-pypackage](#) project template.

2.1 Stable release

To install Cumulus, run this command in your terminal:

```
$ pip install cumulus
```

This is the preferred method to install Cumulus, as it will always install the most recent stable release.

If you don't have [pip](#) installed, this [Python installation guide](#) can guide you through the process.

2.2 From sources

The sources for Cumulus can be downloaded from the [Github repo](#).

You can either clone the public repository:

```
$ git clone git://github.com/brettswift/cumulus
```

Or download the [tarball](#):

```
$ curl -OL https://github.com/brettswift/cumulus/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```


CHAPTER 3

Usage

To use Cumulus in a project:

```
import cumulus
```


- **Pull Requests Welcome**

- use a feature or fix branch: *feature/<name>* or *fix/<name>*
- squash commits
- write a good commit message <https://chris.beams.io/posts/git-commit/>
- Add your name to CONTRIBUTING.rst

4.1 Submit Feedback

The best way to send feedback is to file an issue at <https://github.com/brettswift/cumulus/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 :)

4.1.1 Get Started!

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

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

```
$ git clone git@github.com:your_name_here/cumulus.git
```

3. Install your local copy into a virtualenv. Assuming virtualenv and pyenv are installed,

```
$ pyenv virtualenv 3.6 cumulus36 $ pyenv local cumulus36 $ cd cumulus/ $ python setup.py develop
(sometimes I use pip install -e .)
```

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

```
$ flake8 cumulus tests
$ python setup.py test or py.test
$ tox
```

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

6. Squash changes to a single commit and write a good commit message
 - A good commit message follows: <https://chris.beams.io/posts/git-commit/>
- ```
$ git fetch --all $ git rebase -i [brettsswift]/master $ # use interactive squash to squash all commits (tip:
keep the top one, type s or squash next to all the others
```
7. Push code and Submit a pull request through the GitHub website.

## 4.1.2 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, 3.4, 3.5 and 3.6, and for PyPy. Check [https://travis-ci.org/brettsswift/cumulus/pull\\_requests](https://travis-ci.org/brettsswift/cumulus/pull_requests) and make sure that the tests pass for all supported Python versions.

## 4.1.3 Tips

To run a subset of tests:

```
$ py.test tests.test_cumulus
```

## 4.1.4 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in HISTORY.rst). Then run:

```
$ git checkout -b release/<version>
$ pip install -r requirements_dev.txt # to get bump2version, or just install it_
↳ directly
$ bump2version patch # possible: major / minor / patch
$ git push <brettsswift remote name> release/<version>
$ git push <brettsswift remote name> --tags
```

Travis will then deploy to PyPI if tests pass.

Changelog - release it too!

Once released (or at least when the release commit is on origin/master), generate and push the changelog.

Use this tool: <https://github.com/github-changelog-generator/github-changelog-generator>

You will need a personal access token to github for this: <https://github.com/settings/tokens>

**Note: the documented docker file is broken, but someone has a fix for it here:** `docker run -it -rm -v "$(pwd)":/project markmandel/github-changelog-generator -u brettswift -p cumulus -t <your_token_here>`

The original docker command once fixed, should be used and the temporary one above deleted from this file.

`' docker run -it -rm -v "$(pwd)":/usr/local/src/your-app ferrarimarco/github-changelog-generator'`

Steps:

1. wait for new tag/version to be on master
2. run the docker command above
3. commit and push the changelog in via a Pull Request.



### 5.1 Development Lead

- Brett Swift <brettswift@gmail.com>

### 5.2 Contributors

None yet. Why not be the first?



## CHAPTER 6

---

### Indices and tables

---

- `genindex`
- `modindex`
- `search`