beauty-ocean Documentation

Release 0.1.0

Nick Mavrakis

Jul 16, 2022

Contents

CHAPTER 1

beauty-ocean

Create DigitalOcean droplets like a breeze through the command line.

1.1 Documentation

You may read the full documentation on ReadTheDocs.

1.2 Preparation

Before installing this package make sure you have an account to Digital Ocean and have obtained a Digital Ocean API token The recommended way of storing the token is via environment variable. Once you copy it, do the following:

```
$ vim ~/.profile or nano ~/.profile
# make sure the following line is added
$ export DO_TOKEN="API TOKEN HERE"  # or other name than DO_TOKEN
$ source ~/.profile
```

1.3 Installation

Installation is just a *pip install* away:

```
(virtualenv_name) $ pip install beauty-ocean
```

You are using virtualenv, don't you? If not, install it inside your `~/.local` directory.

```
$ pip install --user beauty-ocean
```

Never ever use sudo!

1.4 Usage

For the moment, this package implements the creation of Digital Ocean droplets but very soon it will support the creation of Domains and Networks. Once installed, the `droplet` command will be available at your disposal. It accepts a single option `--token` or `-t` for short. Defaults to `"DO_TOKEN"` which is the name of the environment variable that you created earlier. If you used a different name then pass that name to the `-t` option.

```
$ droplet

# or
$ droplet -t MY_ENV_NAME_FOR_TOKEN
```

You may, also, pass a file path to the `-t` option where this file holds the token only.

```
$ droplet --t path/to/file/that/holds/the/token
```

Lastly, **but not recommended**, you may pass directly, to the `-t` option, the token itself.

```
$ droplet --t THE_ACTUAL_API_TOKEN_HERE
```

Once the token is resolved, a series of questions will be initiated in order to get the available data from you, submit this data to the Digital Ocean API and create the droplet. All the above come in a good-looking format of questions.

Finally, a json string will be returned with all the droplet data at your disposal to use it in any way you want.

1.5 Features

- Beautiful command-line-interface questions flow with sensible defaults
- Supports remote or local SSH keys addition and/or remote/local Tags

1.6 Demo

An mp4 video demo can be found here.

1.7 Credits

This package was created using:

- Cookiecutter
- audreyr/cookiecutter-pypackage project template
- python-digitalocean python library for DigitalOcean API
- inquirer to ask questions (based on the inquirejs command line UI)
- colored to color the prompt
- yaspin to display a "loading" animation while fetching data
- click to create the command line
- sshpubkeys to parse/validate public key(s)

1.8 Disclaimer

I do not work at DigitalOcean, neither have any benefits (financial or professional) from creating this package. This package was created because it facilitates my workflow during droplet creation and website deployment and I wanted to share it with other developers. Sharing is a good thing!

CHAPTER 2

Installation

2.1 Stable release (preferred method)

To install beauty-ocean, run this command in your terminal (**do not run this command using sudo**, instead create a virtualenv. If you can't create one, install it inside your `~/.local/` dir using `pip install --user beauty-ocean`):

```
(virtualenv_name) $ pip install beauty-ocean
```

This is the preferred method to install `beauty-ocean`, as it will **not** interfere with your system packages and in addition it will always be the most recent stable release.

If you don't have pip installed (which is very unlikely if you are under a virtualenv), this Python installation guide can guide you through the process.

2.2 From sources

The sources for beauty-ocean can be downloaded from the Github repo.

You can either install it directly from the public repository:

```
$ pip install git+https://github.com/manikos/beauty-ocean.git
```

Or download and install the tarball:

```
$ curl -OJL https://github.com/manikos/beauty-ocean/tarball/master
$ pip install manikos-beauty-ocean-<commit_hash>.tar.gz
```

CHAPTER 3

Usage

beauty_ocean was designed to be used not only as a cli command (via the excellent python library click) but also as a function call as well. Assuming that beauty_ocean is already installed inside your virtualenv's dev-dependencies (or globally) then in your terminal:

```
$ droplet

# or
$ droplet --token DIGITAL_OCEAN_API_TOKEN_ENV_NAME

# or
$ droplet --token path/to/file/that/holds/the/token

# or
$ droplet --token THE_ACTUAL_API_TOKEN_HERE

# want help?
$ droplet --help
```

beauty_ocean accepts one option -t (short version) or --token (long version) which can either be:

1. an environment variable name which holds the DigitalOcean API Token (default: DO_TOKEN)
2. a file name (which contains only the DigitalOcean API Token) or
3. the actual DigitalOcean API Token itself

The recommended way to use this tool is to provide an environment variable name to the --token option. If the option is omitted, beauty_ocean will look in your environment variables for DO_TOKEN. If not found, it will raise a ValueError.

The steps that beauty_ocean takes to resolve to an API Token are described below, by priority:

Assuming that you entered: droplet -t ABCDEF then:

1. it will look for an env var named "ABCDEF".
2. fail that, it will look for a file in the current dir named "ABCDEF"

3. fail that, it will use the string `"ABCDEF"` as the DigitalOcean API Token

Once, a valid token is provided then `beauty_ocean` will initiate a list of questions like the droplet's region, image, size, name etc and once all these questions have been answered, a final confirmation dialog will be displayed in order to create the droplet.

Finally, a json string will be returned with all the droplet data at your disposal to use it in any way you want.

3.1 DEMO

I built this tool to enhance automation of Digital Ocean's droplet(s). Future work will include the extension of this tool to automate DNS and Networks.

CHAPTER 4

beauty_ocean

4.1 beauty_ocean package

4.1.1 Subpackages

beauty_ocean.core package

Submodules

beauty_ocean.core.api module

beauty_ocean.core.helpers module

beauty_ocean.core.questions module

Module contents

beauty_ocean.droplet package

Submodules

beauty_ocean.droplet.api module

beauty_ocean.droplet.choices module

beauty_ocean.droplet.entry module

beauty_ocean.droplet.helpers module

beauty_ocean.droplet.questions module

Module contents

4.1.2 Submodules

4.1.3 beauty_ocean.cli module

4.1.4 Module contents

CHAPTER 5

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:

5.1 Types of Contributions

5.1.1 Report Bugs

Report bugs at https://github.com/manikos/beauty_ocean/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.

5.1.2 Fix Bugs

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

5.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

5.1.4 Write Documentation

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

5.1.5 Submit Feedback

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

5.2 Get Started!

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

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

```
$ git clone git@github.com:your_name_here/beauty_ocean.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 beauty_ocean
(beauty_ocean) $ cd beauty_ocean/
(beauty_ocean) $ pip install -e .
```

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 beauty_ocean tests
$ pytest
$ tox
```

To get flake8, tox and pytest, 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.

5.3 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 Python 3.6+ and for PyPy. Check https://travis-ci.org/manikos/beauty_ocean/pull_requests and make sure that the tests pass for all supported Python versions.

5.4 Tips

To run a subset of tests

```
# run all tests inside test_cli.py
$ pytest tests/test_cli.py

# run all tests inside the tests/core/ directory
$ pytest tests/core/

# find all test functions whose names contain "get_account_data"
# and run these tests
$ pytest -k "get_account_data"
```

5.5 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

```
$ bump2version patch # possible: major / minor / patch
$ git push
$ git push --tags
```

Travis will then deploy to PyPI if tests pass.

CHAPTER 6

Credits

6.1 Development Lead

- Nick Mavrakis <mavrakis.n@gmail.com>

6.2 Contributors

None yet. Why not be the first?

CHAPTER 7

History

7.1 0.2.2 (2020-3-28)

- Fix a bug (Session object is not serializable) produced while creating the droplet.

7.2 0.2.1 (2020-1-17)

- Fix a bug (NotFoundError) produced while creating the droplet.

7.3 0.1.4 (2019-04-05)

- Fix exception occurred when token was imported by a file.

7.4 0.1.3 (2018-09-18)

- Fix `create_droplet` function which did not return JSON data. Now it does.

7.5 0.1.2 (2018-09-17)

- Fix Pipenv file and include Pipfile* files into the source distribution.

7.6 0.1.1 (2018-09-16)

- Added check-manifest package and fix bump2version version error.

7.7 0.1.0 (2018-09-15)

- First release on PyPI.

CHAPTER 8

Indices and tables

- genindex
- modindex
- search