giticket Documentation

Release 0.1.0

Milind Shakya

Aug 26, 2022

Contents:

CHAPTER 1

giticket

Auto add ticket info to your git commits.

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

1.1 Features

This hook saves developers time by prepending ticket numbers to commit-msgs. For this to work the following two conditions must be met:

- The ticket format regex specified must match, if the regex is passed in.
- Unless you use `regex_match` mode, the branch name format must be <ticket number>_<rest of the branch name>

For e.g. if you name your branch `JIRA-1234_awesome_feature` and commit `Fix some bug`, the commit will be updated to `JIRA-1234 Fix some bug`.

Pass `--regex=` or update `args: [--regex=<custom regex>]` in your .yaml file if you have custom ticket regex. By default it's `[A-Z]+-\d+`.

Pass `--format=` or update `args: [--format=<custom template string>]` in your .yaml file if you have custom message replacement. By default it's `'{ticket} {commit_msg}`, where `ticket` is replaced with the found ticket number and `commit_msg` is replaced with the original commit message.

Pass `--mode=` or update `args: [--mode=regex_match]` in your .yaml file to extract ticket by the regex rather than relying on branch name convention. With this mode you can also make use of `{tickets}` placeholder in `format` argument value to put multiple comma-separated tickets in the commit message in case your branch contains more than one ticket.

It is best used along with pre-commit. You can use it along with pre-commit by adding the following hook in your `.pre-commit-config.yaml` file.

```
repos:
- repo:  https://github.com/milin/giticket
  rev: v1.3
  hooks:
  - id:  giticket
    args: ['--regex=PROJ-[0-9]', '--format={ticket} {commit_msg}']  # Optional
```

1.2 You need to have precommit setup to use this hook.

Install Pre-commit and the commit-msg hook-type.

```
pip install pre-commit
pre-commit install
pre-commit install --hook-type commit-msg
```

CHAPTER 2

Installation

2.1 Stable release

To install giticket, run this command in your terminal:

```
$ pip install giticket
```

This is the preferred method to install giticket, 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 giticket can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/milin/giticket
```

Or download the tarball:

```
$ curl  -OL https://github.com/milin/giticket/tarball/master
```

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

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

CHAPTER 3

Usage

To use giticket in a project:

```
import giticket
```

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/milin/giticket/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” and “help wanted” is open to whoever wants to implement it.

4.1.3 Implement Features

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

4.1.4 Write Documentation

giticket could always use more documentation, whether as part of the official giticket 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/milin/giticket/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 *giticket* for local development.

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

```
$ git clone git@github.com:your_name_here/giticket.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 giticket
$ cd giticket/
$ python setup.py develop
```

4. Set up pre-commit:

```
$ pip install pre-commit
$ pre-commit install
$ pre-commit install --hook-type commit-msg
```

5. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

6. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 giticket tests      # flake8 will also run on commit
$ python setup.py test or py.test
$ tox
```

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

7. 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
```

8. 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 Python 2.7, 3.4, 3.5 and 3.6, and for PyPy. Check https://travis-ci.org/milin/giticket/pull_requests and make sure that the tests pass for all supported Python versions.

4.4 Tips

To run a subset of tests:

```
$ py.test tests.test_giticket
```

4.5 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:

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

Travis will then deploy to PyPI if tests pass.

CHAPTER 5

History

5.1 0.1.5 (2019-04-23)

- Add custom commit message template to be passed in.

5.2 0.1.0 (2019-01-02)

- First release on PyPI.

CHAPTER 6

Credits

6.1 Development Lead

- Milind Shakya - https://github.com/milin

6.2 Contributors

- Scott Peshak - https://github.com/speshak
- kir0ul - https://github.com/kir0ul
- Daniil Penkin - https://github.com/detouched
- Joseph S. Tate - https://github.com/josephtate

CHAPTER 7

Indices and tables

- genindex
- modindex
- search