pyup Documentation

Release 1.0.2

Jannis Gebauer

Aug 21, 2018

Contents

Contents:

A tool that updates all your project's Python dependency files through Pull Requests on GitHub/GitLab.

CHAPTER 1

About

This repo contains the bot that is running at pyup.io. You can install it locally and run the bot through the command line interface.

Documentation: https://pyup.io/docs/

CHAPTER 2

Installation

To install pyup, run:

```
$ pip install pyupio
```

If you want to update Pipfiles, install the optional pipenv extra:

```
$ pip install dparse[pipenv]
```

CHAPTER 3

Obtain Token

In order to communicate with the github API, you need to create an oauth token for your account:

- Log in to your github account
- Click on settings -> Personal access tokens
- Click on Generate new token
- Make sure to check *repo* and *email* and click on Generate token

CHAPTER 4

Run your first Update

Run:

```
$ pyup --repo=username/repo --user-token=<YOUR_TOKEN> --initial
```

This will check all your requirement files and search for new package versions. If there are updates available, pyup will create a new branch on your repository and create a new commit for every single update. Once all files are up to date, pyup will create a single pull request containing all commits.

Once your repository is up to date and the initial update is merged in, remove the *–initial* flag and run:

```
$ pyup --repo=username/repo --user-token=<YOUR_TOKEN>
```

This will create a new branch and a pull request for every single update. Run a cronjob or a scheduled task somewhere that auto-updates your repository once in a while (e.g. every day) to stay on latest.

Pyup also has experimental support for Gitlab. Generate a personal access token from your profile settings (eg. https://gitlab.com/profile/personal_access_tokens), then run pyup from the cli:

```
# gitlab.com:
$ pyup --provider gitlab --repo=username/repo --user-token=<YOUR_TOKEN>

# other:
$ pyup --provider gitlab --repo=username/repo --user-token=<YOUR_TOKEN>@https://your.
↪gitlab/
```

CHAPTER 5

Installation

To install pyup, run:

```
$ pip install pyupio
$ pip install -e git+https://github.com/jayfk/PyGithub.git@top#egg=PyGithub
```

CHAPTER 6

Usage

To use pyup in a project:

```
import pyup
```

CHAPTER 7

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:

7.1 Types of Contributions

7.1.1 Report Bugs

Report bugs at https://github.com/pyupio/pyup/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.

7.1.2 Fix Bugs

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

7.1.3 Implement Features

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

7.1.4 Write Documentation

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

7.1.5 Submit Feedback

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

7.2 Get Started!

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

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

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

7.3 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 2.6, 2.7, 3.3, and 3.4, and for PyPy. Check https://travis-ci.org/pyupio/pyup/pull_requests and make sure that the tests pass for all supported Python versions.

7.4 Tips

To run a subset of tests:

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

CHAPTER 8

Credits

8.1 Development Lead

- Jannis Gebauer <ja.geb@me.com>

8.2 Contributors

None yet. Why not be the first?

CHAPTER 9

History

CHAPTER 10

1.0.2 (2018-8-21)

- Order the hashes being updated on requirements files.

CHAPTER 11

1.0.1 (2018-4-17)

- The previous release contained a bug that caused the build system to deploy the wrong commit to PyPi.

CHAPTER 12

1.0.0 (2018-4-17)

- Added new config options for GitLab (thanks @kairichard)

CHAPTER 13

0.11.0 (2018-4-6)

- Pipenv is now an optional transitive dependency. If you want to update Pipfiles, install it with dparse[pipenv]
- Hashin is now no longer a dependency
- The bot uses the new pypi.org now
- Creating issues on invalid config files is now configurable

CHAPTER 14

0.10.0 (2018-3-15)

- The bot now creates issues if there are any problems with the config file
- Added support for setup.cfg files (thanks @kxepal)
- Switched to the GitLab v4 API (thanks @kxepal)
- Fixed a template error (thanks @kxepal)

CHAPTER 15

0.9.0 (2018-3-01)

- Added a new update filter that allows to restrict patch/minor updates
- Added a new filter extension that allows to specify a date on which the filter expires
- Dropped support for Python 2.6 (if this ever worked)
- Added experimental support for Pipfiles and Pipfiles.lock
- The bot now correctly sets the date in monthly pull requests
- Whitespaces in filter comments should no longer be significant
- Fixed a minor bug that occured with private packages

CHAPTER 16

0.8.1 (2017-7-28)

- Fixed another packaging error.

CHAPTER 17

0.8.1 (2017-7-25)

- Fixed a packaging error where not all template files were included.

CHAPTER 18

0.8.0 (2017-7-20)

- This release adds support for insecure packages and pull requests with attached changelogs.

CHAPTER 19

0.7.0 (2017-7-13)

- Fixed a bug on the CLI that prevented hashed requirements to be parsed correctly
- Switched to the new dparse library, adding experimental support for tox and conda files.
- Added support for GitHubs new collaborator invitation system.
- The bot now correctly parses requirement files that begin with a whitespace.
- Fixed a bug with requirement files that had special characters in the filepath.
- Overall improvements with hashed requirement files. Almost all flavors should now be parsed correctly
- Added support for Gitlab, thanks a lot to @samdroid-apps
- Added support for compatible releases

CHAPTER 20

0.6.0 (2017-2-1)

- Fixed the CLI, it should be working again
- Now supports GitHub Integrations (experimental)
- Added new config: PR prefixes, branch prefixes
- Fixed an error not correclty formatting whitespace
- Added support for hashed requirement files
- The bot is now able to write config files to the repo
- Support for environment markers in requirements has been added
- It's now possible to have finer grained control over what's being updated.

CHAPTER 21

0.5.0 (2016-10-21)

- The bot now parses requirement extras correctly
- Made the config parser more robust
- Fixed a possible endless loop on conflicting PRs
- Added schedules to the config parser
- Now using PyGithub again

CHAPTER 22

0.4.0 (2016-8-30)

- Added a new feature: The bot can now add a label to pull requests.

CHAPTER 23

0.3.0 (2016-7-28)

- Fixed a bug where a race condition occurred when committing too fast.
- Various parser enhancements
- Empty commits are now filtered out automatically
- The bot now supports custom branches and custom index servers
- Stale pull requests will now be closed automatically
- Switched to setuptools new Requirement implementation
- Enhanced logging
- A lot of smaller bugfixes

CHAPTER 24

0.2.0 (2016-1-7)

- Added advanced filtering options

CHAPTER 25

0.1.4 (2015-12-30)

- Fixed a bug with the github provider when committing too fast.
- Requirement content replace function had a bug where not always the right requirement

was replaced

CHAPTER 26

0.1.3 (2015-12-27)

- PyGithub should be installed as a specific dependency to keep things sane and simple until the changes on upstream are merged.

CHAPTER 27

0.1.2 (2015-12-27)

- Use development version of pygithub.

CHAPTER 28

0.1.1 (2015-12-27)

- Fixed minor packing issue.

CHAPTER 29

0.1 (2015-12-27)

- (silent) release on PyPI.

CHAPTER 30

Indices and tables

- genindex
- modindex
- search