GitPython 3.1.32

`pip install GitPython`

Latest version

Released: Jul 10, 2023

GitPython is a Python library used to interact with Git repositories

Statistics

GitHub statistics:

- **Stars:** 4001
- **Forks:** 857
- **Open issues:** 142
- **Open PRs:** 2

View statistics for this project via Libraries.io, or by using our public dataset on Google BigQuery

Meta

License: BSD License (BSD)

Author: Sebastian Thiel, Michael Trier

Requires: Python >=3.7

Maintainers

ByronBates

mtrier

nvie

Sebastian.Thiel

yarikoptic

Classifiers

Development Status
- 5 - Production/Stable

Environment
- Console

Intended Audience
- Developers

License
- OSI Approved :: BSD License

Operating System
- MacOS :: MacOS X
- Microsoft :: Windows
- OS Independent
- POSIX

Programming Language
- Python
- Python :: 3
- Python :: 3.7
- Python :: 3.8
- Python :: 3.9
- Python :: 3.10
- Python :: 3.11

Typing
- Typed

Project description

Python package passing | docs passing | in repositories 27

Gitoxide: A peek into the future…

I started working on GitPython in 2009, back in the days when Python was 'my thing' and I had great plans with it. Of course, back in the days, I didn't really know what I was doing and this shows in many places. Somewhat similar to Python this happens to be 'good enough', but at the same time is deeply flawed and broken beyond repair.

By now, GitPython is widely used and I am sure there is a good reason for that, it's something to be proud of and happy about. The community is maintaining the software and is keeping it relevant for which I am absolutely grateful. For the time to come I am happy to continue maintaining GitPython, remaining hopeful that one day it won't be needed anymore.

More than 15 years after my first meeting with 'git' I am still in excited about it, and am happy to finally have the tools and probably the skills to scratch that itch of mine: implement `git` in a way that makes tool creation a piece of cake for most.

If you like the idea and want to learn more, please head over to gitoxide, an implementation of 'git' in Rust.

GitPython

GitPython is a python library used to interact with git repositories, high-level like git-porcelain, or low-level like git-plumbing.

It provides abstractions of git objects for easy access of repository data often backed by calling the `git` command-line program.

DEVELOPMENT STATUS

This project is in **maintenance mode**, which means that

- …there will be no feature development, unless these are contributed
- …there will be no bug fixes, unless they are relevant to the safety of users, or contributed
- …issues will be responded to with waiting times of up to a month

The project is open to contributions of all kinds, as well as new maintainers.

REQUIREMENTS

GitPython needs the `git` executable to be installed on the system and available in your `PATH` for most operations. If it is not in your `PATH`, you can help GitPython find it by setting the `GIT_PYTHON_GIT_EXECUTABLE=<path/to/git>` environment variable.

- Git (1.7.x or newer)
- Python >= 3.7

The list of dependencies are listed in `./requirements.txt` and `./test-requirements.txt`. The installer takes care of installing them for you.

INSTALL

If you have downloaded the source code:

```
python setup.py install
```

or if you want to obtain a copy from the Pypi repository:

```
pip install GitPython
```

Both commands will install the required package dependencies.

A distribution package can be obtained for manual installation at: http://pypi.python.org/pypi/GitPython.

If you like to clone from source, you can do it like so:

```
git clone https://github.com/gitpython-developers/G
git submodule update --init --recursive
./init-tests-after-clone.sh
```

Limitations

Leakage of System Resources

GitPython is not suited for long-running processes (like daemons) as it tends to leak system resources. It was written in a time where destructors (as implemented in the `__del__` method) still ran deterministically.

In case you still want to use it in such a context, you will want to search the codebase for `__del__` implementations and call these yourself when you see fit.

Another way assure proper cleanup of resources is to factor out GitPython into a separate process which can be dropped periodically.

Windows support

See Issue #525.

RUNNING TESTS

Important: Right after cloning this repository, please be sure to have executed the `./init-tests-after-clone.sh` script in the repository root. Otherwise you will encounter test failures.

On *Windows*, make sure you have `git-daemon` in your PATH. For MINGW-git, the `git-daemon.exe` exists in `Git\mingw64\libexec\git-core\`; CYGWIN has no daemon, but should get along fine with MINGW's.

Ensure testing libraries are installed. In the root directory, run: `pip install -r test-requirements.txt`

To lint, run: `pre-commit run --all-files`

To typecheck, run: `mypy -p git`

To test, run: `pytest`

Configuration for flake8 is in the ./.flake8 file.

Configurations for mypy, pytest and coverage.py are in ./pyproject.toml.

The same linting and testing will also be performed against different supported python versions upon submitting a pull request (or on each push if you have a fork with a "main" branch and actions enabled).

Contributions

Please have a look at the contributions file.

INFRASTRUCTURE

- User Documentation
- Questions and Answers
- Please post on stackoverflow and use the `gitpython` tag
- Issue Tracker
  - Post reproducible bugs and feature requests as a new issue. Please be sure to provide the following information if posting bugs:
    - GitPython version (e.g. `import git; git.__version__`)
    - Python version (e.g. `python --version`)
    - The encountered stack-trace, if applicable
    - Enough information to allow reproducing the issue

How to make a new release

- Update/verify the **version** in the `VERSION` file
- Update/verify that the `doc/source/changes.rst` changelog file was updated
- Commit everything
- Run `git tag -s <version>` to tag the version in Git
- Run `make release`
- Close the milestone mentioned in the *changelog* and create a new one. *Do not reuse milestones by renaming them*.
- set the upcoming version in the `VERSION` file, usually be incrementing the patch level, and possibly by appending `-dev`. Probably you want to `git push` once more.

How to verify a release (DEPRECATED)

Note that what follows is deprecated and future releases won't be signed anymore. More details about how it came to that can be found in this issue.

Please only use releases from `pypi` as you can verify the respective source tarballs.

This script shows how to verify the tarball was indeed created by the authors of this project:

```
curl https://files.pythonhosted.org/packages/09/bc/
curl https://files.pythonhosted.org/packages/09/bc/
gpg --verify gitpython-signature.asc gitpython.whl
```

which outputs

```
gpg: Signature made Fr  4 Sep 10:04:50 2020 CST
gpg:                using RSA key 27C50E7F590947D72
gpg: Good signature from "Sebastian Thiel (YubiKey
gpg:                 aka "Sebastian Thiel (In Rust
```

You can verify that the keyid indeed matches the release-signature key provided in this repository by looking at the keys details:

```
gpg --list-packets ./release-verification-key.asc
```

You can verify that the commit adding it was also signed by it using:

```
git show --show-signature  ./release-verification-k
```

If you would like to trust it permanently, you can import and sign it:

```
gpg --import ./release-verification-key.asc
gpg --edit-key 4C08421980C9

> sign
> save
```

Projects using GitPython

- PyDriller
- Kivy Designer
- Prowl
- Python Taint
- Buster
- git-ftp
- Git-Pandas
- PyGitUp
- PyJFuzz
- Loki
- Omniwallet
- GitViper
- Git Gud

LICENSE

New BSD License. See the LICENSE file.