Twine Documentation

Release 3.4.2.dev1+geff3a45

Donald Stufft and individual contributors

Apr 10, 2021

Further documentation

Twine is a utility for publishing Python packages on PyPI.

It provides build system independent uploads of source and binary distribution artifacts for both new and existing projects.

Table of Contents

CHAPTER 1

Why Should I Use This?

The goal of Twine is to improve PyPI interaction by improving security and testability.

The biggest reason to use Twine is that it securely authenticates you to PyPI over HTTPS using a verified connection, regardless of the underlying Python version. Meanwhile, `python setup.py upload` will only work correctly and securely if your build system, Python version, and underlying operating system are configured properly.

Secondly, Twine encourages you to build your distribution files. `python setup.py upload` only allows you to upload a package as a final step after building with `distutils` or `setuptools`, within the same command invocation. This means that you cannot test the exact file you're going to upload to PyPI to ensure that it works before uploading it.

Finally, Twine allows you to pre-sign your files and pass the `.asc` files into the command line invocation (`twine upload myproject-1.0.1.tar.gz myproject-1.0.1.tar.gz.asc`). This enables you to be assured that you're typing your `gpg` passphrase into `gpg` itself and not anything else, since *you* will be the one directly executing `gpg --detach-sign -a <filename>`.

CHAPTER 2

Features

- Verified HTTPS connections
- Uploading doesn't require executing `setup.py`
- Uploading files that have already been created, allowing testing of distributions before release
- Supports uploading any packaging format (including wheels)

CHAPTER 3

Installation

```
$ pip install twine
```

CHAPTER 4

Using Twine

1. Create some distributions in the normal way:

```
$ python setup.py sdist bdist_wheel
```

2. Upload with `twine` to Test PyPI and verify things look right. Twine will automatically prompt for your username and password:

```
$ twine upload -r testpypi dist/*
username: ...
password:
...
```

3. Upload to PyPI:

```
$ twine upload dist/*
```

4. Done!

More documentation on using Twine to upload packages to PyPI is in the Python Packaging User Guide.

CHAPTER 5

Commands

5.1 twine upload

Uploads one or more distributions to a repository.

```
$ twine upload -h
usage: twine upload [-h] [-r REPOSITORY] [--repository-url REPOSITORY_URL]
                    [-s] [--sign-with SIGN_WITH] [-i IDENTITY] [-u USERNAME]
                    [-p PASSWORD] [-c COMMENT] [--config-file CONFIG_FILE]
                    [--skip-existing] [--cert path] [--client-cert path]
                    [--verbose] [--disable-progress-bar]
                    dist [dist ...]

positional arguments:
  dist                  The distribution files to upload to the repository
                        (package index). Usually dist/* . May additionally
                        contain a .asc file to include an existing signature
                        with the file upload.

optional arguments:
  -h, --help            show this help message and exit
  -r REPOSITORY, --repository REPOSITORY
                        The repository (package index) to upload the package
                        to. Should be a section in the config file (default:
                        pypi). (Can also be set via TWINE_REPOSITORY
                        environment variable.)
  --repository-url REPOSITORY_URL
                        The repository (package index) URL to upload the
                        package to. This overrides --repository. (Can also be
                        set via TWINE_REPOSITORY_URL environment variable.)
  -s, --sign            Sign files to upload using GPG.
  --sign-with SIGN_WITH
                        GPG program used to sign uploads (default: gpg).
  -i IDENTITY, --identity IDENTITY
```

(continues on next page)

(continued from previous page)

```
                        GPG identity used to sign files.
  -u USERNAME, --username USERNAME
                        The username to authenticate to the repository
                        (package index) as. (Can also be set via
                        TWINE_USERNAME environment variable.)
  -p PASSWORD, --password PASSWORD
                        The password to authenticate to the repository
                        (package index) with. (Can also be set via
                        TWINE_PASSWORD environment variable.)
  --non-interactive     Do not interactively prompt for username/password
                        if the required credentials are missing. (Can also
                        be set via TWINE_NON_INTERACTIVE environment
                        variable.)
  -c COMMENT, --comment COMMENT
                        The comment to include with the distribution file.
  --config-file CONFIG_FILE
                        The .pypirc config file to use.
  --skip-existing       Continue uploading files if one already exists. (Only
                        valid when uploading to PyPI. Other implementations
                        may not support this.)
  --cert path           Path to alternate CA bundle (can also be set via
                        TWINE_CERT environment variable).
  --client-cert path    Path to SSL client certificate, a single file
                        containing the private key and the certificate in PEM
                        format.
  --verbose             Show verbose output.
  --disable-progress-bar
                        Disable the progress bar.
```

5.2 twine check

Checks whether your distribution's long description will render correctly on PyPI.

```
$ twine check -h
usage: twine check [-h] [--strict] dist [dist ...]

positional arguments:
  dist        The distribution files to check, usually dist/*

optional arguments:
  -h, --help  show this help message and exit
  --strict    Fail on warnings
```

5.3 twine register

WARNING: The `register` command is no longer necessary if you are uploading to pypi.org. As such, it is no longer supported in Warehouse (the new PyPI software running on pypi.org). However, you may need this if you are using a different package index.

For completeness, its usage:

```
$ twine register -h

usage: twine register [-h] -r REPOSITORY [--repository-url REPOSITORY_URL]
                      [-u USERNAME] [-p PASSWORD] [-c COMMENT]
                      [--config-file CONFIG_FILE] [--cert path]
                      [--client-cert path]
                      package

positional arguments:
  package               File from which we read the package metadata.

optional arguments:
  -h, --help            show this help message and exit
  -r REPOSITORY, --repository REPOSITORY
                        The repository (package index) to register the package
                        to. Should be a section in the config file. (Can also
                        be set via TWINE_REPOSITORY environment variable.)
                        Initial package registration no longer necessary on
                        pypi.org:
                        https://packaging.python.org/guides/migrating-to-pypi-
                        org/
  --repository-url REPOSITORY_URL
                        The repository (package index) URL to register the
                        package to. This overrides --repository. (Can also be
                        set via TWINE_REPOSITORY_URL environment variable.)
  -u USERNAME, --username USERNAME
                        The username to authenticate to the repository
                        (package index) as. (Can also be set via
                        TWINE_USERNAME environment variable.)
  -p PASSWORD, --password PASSWORD
                        The password to authenticate to the repository
                        (package index) with. (Can also be set via
                        TWINE_PASSWORD environment variable.)
  --non-interactive     Do not interactively prompt for username/password
                        if the required credentials are missing. (Can also
                        be set via TWINE_NON_INTERACTIVE environment
                        variable.)
  -c COMMENT, --comment COMMENT
                        The comment to include with the distribution file.
  --config-file CONFIG_FILE
                        The .pypirc config file to use.
  --cert path           Path to alternate CA bundle (can also be set via
                        TWINE_CERT environment variable).
  --client-cert path    Path to SSL client certificate, a single file
                        containing the private key and the certificate in PEM
                        format.
```

CHAPTER 6

Configuration

Twine can read repository configuration from a `.pypirc` file, either in your home directory, or provided with the `--config-file` option. For details on writing and using `.pypirc`, see the specification in the Python Packaging User Guide.

6.1 Environment Variables

Twine also supports configuration via environment variables. Options passed on the command line will take precedence over options set via environment variables. Definition via environment variable is helpful in environments where it is not convenient to create a `.pypirc` file (for example, on a CI/build server).

- `TWINE_USERNAME` - the username to use for authentication to the repository.
- `TWINE_PASSWORD` - the password to use for authentication to the repository.
- `TWINE_REPOSITORY` - the repository configuration, either defined as a section in `.pypirc` or provided as a full URL.
- `TWINE_REPOSITORY_URL` - the repository URL to use.
- `TWINE_CERT` - custom CA certificate to use for repositories with self-signed or untrusted certificates.
- `TWINE_NON_INTERACTIVE` - Do not interactively prompt for username/password if the required credentials are missing.

CHAPTER 7

Keyring Support

Instead of typing in your password every time you upload a distribution, Twine allows storing a username and password securely using keyring. Keyring is installed with Twine but for some systems (Linux mainly) may require additional installation steps.

Once Twine is installed, use the `keyring` program to set a username and password to use for each package index (repository) to which you may upload.

For example, to set a username and password for PyPI:

```
$ keyring set https://upload.pypi.org/legacy/ your-username
```

or

```
$ python3 -m keyring set https://upload.pypi.org/legacy/ your-username
```

and enter the password when prompted.

For a different repository, replace the URL with the relevant repository URL. For example, for Test PyPI, use `https://test.pypi.org/legacy/`.

The next time you run `twine`, it will prompt you for a username and will grab the appropriate password from the keyring.

Note: If you are using Linux in a headless environment (such as on a server) you'll need to do some additional steps to ensure that Keyring can store secrets securely. See Using Keyring on headless systems.

7.1 Disabling Keyring

In most cases, simply not setting a password with `keyring` will allow Twine to fall back to prompting for a password. In some cases, the presence of Keyring will cause unexpected or undesirable prompts from the backing system. In these cases, it may be desirable to disable Keyring altogether. To disable Keyring, simply invoke:

```
$ keyring --disable
```

or

```
$ python -m keyring --disable
```

That command will configure for the current user the "null" keyring, effectively disabling the functionality, and allowing Twine to prompt for passwords.

See twine 338 for discussion and background.

CHAPTER 8

Resources

- IRC: `#pypa` on irc.freenode.net
- GitHub repository
- User and developer documentation
- Python Packaging User Guide

CHAPTER 9

Contributing

See our developer documentation for how to get started, an architectural overview, and our future development plans.

CHAPTER 10

Code of Conduct

Everyone interacting in the Twine project's codebases, issue trackers, chat rooms, and mailing lists is expected to follow the PSF Code of Conduct.

10.1 Contributing

We are happy you have decided to contribute to twine.

Please see the GitHub repository for code and more documentation, and the official Python Packaging User Guide for user documentation. You can also join `#pypa` or `#pypa-dev` on Freenode, or the distutils-sig mailing list, to ask questions or get involved.

10.1.1 Getting started

We recommend you use a virtual environment, so that twine and its dependencies do not interfere with other packages installed on your machine.

Clone the twine repository from GitHub, then make and activate a virtual environment that uses Python 3.6 or newer as the default Python. For example:

```
cd /path/to/your/local/twine
python3.6 -m venv venv
source venv/bin/activate
```

Then, run the following command:

```
pip install -e .
```

Now, in your virtual environment, `twine` is pointing at your local copy, so when you make changes, you can easily see their effect.

We use tox to run tests, check code style, and build the documentation. To install `tox` in your active virtual environment, run:

```
pip install tox
```

Building the documentation

Additions and edits to twine's documentation are welcome and appreciated.

To preview the docs while you're making changes, run:

```
tox -e watch-docs
```

Then open a web browser to http://127.0.0.1:8000.

When you're done making changes, lint and build the docs locally before making a pull request. In your active virtual environment, run:

```
tox -e docs
```

The HTML of the docs will be written to `docs/_build/html`.

Code style

To automatically reformat your changes with isort and black, run:

```
tox -e format
```

To detect any remaining code smells with flake8, run:

```
tox -e lint
```

To perform strict type-checking using mypy, run:

```
tox -e types
```

Any errors from `lint` or `types` need to be fixed manually.

Additionally, we prefer that `import` statements be used for packages and modules only, rather than individual classes or functions.

Testing

We use pytest for writing and running tests.

To run the tests in your virtual environment, run:

```
tox -e py
```

To pass options to `pytest`, e.g. the name of a test, run:

```
tox -e py -- tests/test_upload.py::test_exception_for_http_status
```

Twine is continuously tested against Python 3.6, 3.7, 3.8, and 3.9 using GitHub Actions. To run the tests against a specific version, e.g. Python 3.6, you will need it installed on your machine. Then, run:

```
tox -e py36
```

To run the "integration" tests of uploading to real package indexes, run:

```
tox -e integration
```

To run the tests against all supported Python versions, check code style, and build the documentation, run:

```
tox
```

10.1.2 Submitting changes

1. Fork the GitHub repository.
2. Make a branch off of `master` and commit your changes to it.
3. Run the tests, check code style, and build the docs as described above.
4. Optionally, add your name to the end of the `AUTHORS` file using the format `Name <email@domain.com> (url)`, where the `(url)` portion is optional.
5. Submit a pull request to the `master` branch on GitHub, referencing an open issue.
6. Add a changelog entry.

Changelog entries

The `docs/changelog.rst` file is built by towncrier from files in the `changelog/` directory. To add an entry, create a file in that directory named `{number}.{type}.rst`, where `{number}` is the pull request number, and `{type}` is `feature`, `bugfix`, `doc`, `removal`, or `misc`.

For example, if your PR number is 1234 and it's fixing a bug, then you would create `changelog/1234.bugfix.rst`. PRs can span multiple categories by creating multiple files: if you added a feature and deprecated/removed an old feature in PR #5678, you would create `changelog/5678.feature.rst` and `changelog/5678.removal.rst`.

A changelog entry is meant for end users and should only contain details relevant to them. In order to maintain a consistent style, please keep the entry to the point, in sentence case, shorter than 80 characters, and in an imperative tone. An entry should complete the sentence "This change will ...". If one line is not enough, use a summary line in an imperative tone, followed by a description of the change in one or more paragraphs, each wrapped at 80 characters and separated by blank lines.

You don't need to reference the pull request or issue number in a changelog entry, since towncrier will add a link using the number in the file name, and the pull request should reference an issue number. Similarly, you don't need to add your name to the entry, since that will be associated with the pull request.

Changelog entries are rendered using reStructuredText, but they should only have minimal formatting (such as ``monospaced text``).

10.1.3 Architectural overview

Twine is a command-line tool for interacting with PyPI securely over HTTPS. Its three purposes are to be:

1. A user-facing tool for publishing on pypi.org
2. A user-facing tool for publishing on other Python package indexes (e.g., `devpi` instances)
3. A useful API for other programs (e.g., `zest.releaser`) to call for publishing on any Python package index

Currently, twine has two principle functions: uploading new packages and registering new projects (`register` is no longer supported on PyPI, and is in Twine for use with other package indexes).

Its command line arguments are parsed in `twine/cli.py`. The code for registering new projects is in `twine/commands/register.py`, and the code for uploading is in `twine/commands/upload.py`. The file `twine/package.py` contains a single class, `PackageFile`, which hashes the project files and extracts their metadata. The file `twine/repository.py` contains the `Repository` class, whose methods control the URL the package is uploaded to (which the user can specify either as a default, in the `.pypirc` file, or pass on the command line), and the methods that upload the package securely to a URL.

Where Twine gets configuration and credentials

A user can set the repository URL, username, and/or password via command line, `.pypirc` files, environment variables, and `keyring`.

10.1.4 Adding a maintainer

A checklist for adding a new maintainer to the project.

1. Add them as a Member in the GitHub repo settings.
2. Get them Test PyPI and canon PyPI usernames and add them as a Maintainer on our Test PyPI project and canon PyPI.

10.1.5 Making a new release

A checklist for creating, testing, and distributing a new version.

1. Choose a version number, e.g. `VERSION=3.3.0`.
2. Create a new branch, e.g. `git switch -c release-$VERSION`.
3. Run `tox -e changelog -- --version $VERSION` to build `docs/changelog.rst`.
4. Commit and open a pull request for review.
5. Merge the pull request, and ensure the GitHub Actions build passes.
6. Create a new git tag with `git tag -m "Release v$VERSION" $VERSION`.
7. Push the new tag with `git push upstream $VERSION`.
8. Watch the release in GitHub Actions.
9. Send announcement email to distutils-sig mailing list and celebrate.

10.1.6 Future development

See our open issues.

In the future, `pip` and `twine` may merge into a single tool; see ongoing discussion.

10.2 Changelog

This project follows the semantic versioning and pre-release versioning schemes recommended by the Python Packaging Authority.

10.2.1 3.4.1 (2021-03-16)

Bugfixes

- Fix a regression that was causing some namespace packages with dots in them fail to upload to PyPI. (#745)

10.2.2 3.4.0 (2021-03-15)

Features

- Prefer importlib.metadata for entry point handling. (#728)
- Rely on importlib_metadata 3.6 for nicer entry point processing. (#732)
- Eliminate dependency on setuptools/pkg_resources and replace with packaging and importlib_metadata. (#736)

10.2.3 3.3.0 (2020-12-23)

Features

- Print files to be uploaded using `upload --verbose` (#670)
- Print configuration file location when using `upload --verbose` (#675)
- Print source and values of credentials when using `upload --verbose` (#685)
- Add support for Python 3.9 (#708)
- Turn warnings into errors when using `check --strict` (#715)

Bugfixes

- Make password optional when using `upload --client-cert` (#678)
- Support more Nexus versions with `upload --skip-existing` (#693)
- Support Gitlab Enterprise with `upload --skip-existing` (#698)
- Show a better error message for malformed files (#714)

Improved Documentation

- Adopt PSF code of conduct (#680)
- Adopt towncrier for the changleog (#718)

10.2.4 3.2.0 (2020-06-24)

Features

- Improve display of HTTP errors during upload (#666)
- Print packages and signatures to be uploaded when using `--verbose` option (#652)
- Use red text when printing errors on the command line (#649)

- Require repository URL scheme to be `http` or `https` (#602)
- Add type annotations, checked with mypy, with **PEP 561** support for users of Twine's API (#231)

Bugfixes

- Update URL to `.pypirc` specification (#655)
- Don't raise an exception when Python version can't be parsed from filename (#612)
- Fix inaccurate retry message during `upload` (#611)
- Clarify error messages for archive format (#601)

10.2.5 3.1.1 (2019-11-27)

Bugfixes

- Restore `--non-interactive` as a flag not expecting an argument. (#548)

10.2.6 3.1.0 (2019-11-23)

Features

- Add support for specifying `--non-interactive` as an environment variable. (#547)

10.2.7 3.0.0 (2019-11-18)

Features

- When a client certificate is indicated, all password processing is disabled. (#336)
- Add `--non-interactive` flag to abort upload rather than interactively prompt if credentials are missing. (#489)
- Twine now unconditionally requires the keyring library and no longer supports uninstalling `keyring` as a means to disable that functionality. Instead, use `keyring --disable` keyring functionality if necessary. (#524)
- Add Python 3.8 to classifiers. (#518)

Bugfixes

- More robust handling of server response in `--skip-existing` (#332)

10.2.8 2.0.0 (2019-09-24)

Features

- Twine now requires Python 3.6 or later. Use pip 9 or pin to "twine<2" to install twine on older Python versions. (#437)

Bugfixes

- Require requests 2.20 or later to avoid reported security vulnerabilities in earlier releases. (#491)

10.2.9 1.15.0 (2019-09-17)

Features

- Improved output on `check` command: Prints a message when there are no distributions given to check. Improved handling of errors in a distribution's markup, avoiding messages flowing through to the next distribution's errors. (#488)

10.2.10 1.14.0 (2019-09-06)

Features

- Show Warehouse URL after uploading a package (#459)
- Better error handling and gpg2 fallback if gpg not available. (#456)
- Now provide a more meaningful error on redirect during upload. (#310)

Bugfixes

- Fail more gracefully when encountering bad metadata (#341)

10.2.11 1.13.0 (2019-02-13)

Features

- Add disable_progress_bar option to disable tqdm. (#427)
- Allow defining an empty username and password in .pypirc. (#426)
- Support keyring.get_credential. (#419)
- Support keyring.get_username_and_password. (#418)
- Add Python 3.7 to classifiers. (#416)

Bugfixes

- Restore prompts while retaining support for suppressing prompts. (#452)
- Avoid requests-toolbelt to 0.9.0 to prevent attempting to use openssl when it isn't available. (#447)
- Use io.StringIO instead of StringIO. (#444)
- Only install pyblake2 if needed. (#441)
- Use modern Python language features. (#436)
- Specify python_requires in setup.py (#435)
- Use https URLs everywhere. (#432)

- Fix –skip-existing for Nexus Repos. (#428)
- Remove unnecessary usage of readme_render.markdown. (#421)
- Don't crash if there's no package description. (#412)
- Fix keyring support. (#408)

Misc

- Refactor tox env and travis config. (#439)

10.2.12 1.12.1 (2018-09-24)

Bugfixes

- Fix regression with upload exit code (#404)

10.2.13 1.12.0 (2018-09-24)

Features

- Add `twine check` command to check long description (#395)
- Drop support for Python 3.3 (#392)
- Empower `--skip-existing` for Artifactory repositories (#363)

Bugfixes

- Avoid MD5 when Python is compiled in FIPS mode (#367)

10.2.14 1.11.0 (2018-03-19)

Features

- Remove PyPI as default `register` package index. (#320)
- Support Metadata 2.1 (**PEP 566**), including Markdown for `description` fields. (#319)

Bugfixes

- Raise exception if attempting upload to deprecated legacy PyPI URLs. (#322)
- Avoid uploading to PyPI when given alternate repository URL, and require `http://` or `https://` in `repository_url`. (#269)

Misc

- Update PyPI URLs. (#318)
- Add new maintainer, release checklists. (#314)
- Add instructions on how to use keyring. (#277)

10.2.15 1.10.0 (2018-03-07)

Features

- Link to changelog from `README` (#46)
- Reorganize & improve user & developer documentation. (#304)
- Revise docs predicting future of `twine` (#303)
- Add architecture overview to docs (#296)
- Add doc building instructions (#295)
- Declare support for Python 3.6 (#257)
- Improve progressbar (#256)

Bugfixes

- Degrade gracefully when keyring is unavailable (#315)
- Fix changelog formatting (#299)
- Fix syntax highlighting in `README` (#298)
- Fix Read the Docs, tox, Travis configuration (#297)
- Fix Travis CI and test configuration (#286)
- Print progress to `stdout`, not `stderr` (#268)
- Fix `--repository[-url]` help text (#265)
- Remove obsolete registration guidance (#200)

10.2.16 1.9.1 (2017-05-27)

Bugfixes

- Blacklist known bad versions of Requests. (#253)

10.2.17 1.9.0 (2017-05-22)

Bugfixes

- Twine sends less information about the user's system in the User-Agent string. (#229)
- Fix `--skip-existing` when used to upload a package for the first time. (#220)
- Fix precedence of `--repository-url` over `--repository`. (#206)

Misc

- Twine will now resolve passwords using the keyring if available. Module can be required with the `keyring` extra.
- Twine will use `hashlib.blake2b` on Python 3.6+ instead of pyblake2

10.2.18 1.8.1 (2016-08-09)

Misc

- Check if a package exists if the URL is one of:
  - `https://pypi.python.org/pypi/`
  - `https://upload.pypi.org/`
  - `https://upload.pypi.io/`

  This helps people with `https://upload.pypi.io` still in their `.pypirc` file.

10.2.19 1.8.0 (2016-08-08)

Features

- Switch from upload.pypi.io to upload.pypi.org. (#201)
- Retrieve configuration from the environment as a default. (#144)
  - Repository URL will default to `TWINE_REPOSITORY`
  - Username will default to `TWINE_USERNAME`
  - Password will default to `TWINE_PASSWORD`
- Allow the Repository URL to be provided on the command-line (`--repository-url`) or via an environment variable (`TWINE_REPOSITORY_URL`). (#166)
- Generate Blake2b 256 digests for packages *if* `pyblake2` is installed. Users can use `python -m pip install twine[with-blake2]` to have `pyblake2` installed with Twine. (#171)

Misc

- Generate SHA256 digest for all packages by default.
- Stop testing on Python 2.6.
- Warn users if they receive a 500 error when uploading to `*pypi.python.org` (#199)

10.2.20 1.7.4 (2016-07-09)

Bugfixes

- Correct a packaging error.

10.2.21 1.7.3 (2016-07-08)

Bugfixes

- Fix uploads to instances of pypiserver using `--skip-existing`. We were not properly checking the return status code on the response after attempting an upload. (#195)

Misc

- Avoid attempts to upload a package if we can find it on Legacy PyPI.

10.2.22 1.7.2 (2016-07-05)

Bugfixes

- Fix issue where we were checking the existence of packages even if the user didn't specify `--skip-existing`. (#189) (#191)

10.2.23 1.7.1 (2016-07-05)

Bugfixes

- Clint was not specified in the wheel metadata as a dependency. (#187)

10.2.24 1.7.0 (2016-07-04)

Features

- Support `--cert` and `--client-cert` command-line flags and config file options for feature parity with pip. This allows users to verify connections to servers other than PyPI (e.g., local package repositories) with different certificates. (#142)
- Add progress bar to uploads. (#152)
- Allow `--skip-existing` to work for 409 status codes. (#162)
- Implement retries when the CDN in front of PyPI gives us a 5xx error. (#167)
- Switch Twine to upload to pypi.io instead of pypi.python.org. (#177)

Bugfixes

- Allow passwords to have `%s` in them. (#186)

10.2.25 1.6.5 (2015-12-16)

Bugfixes

- Bump requests-toolbelt version to ensure we avoid ConnectionErrors (#155)

10.2.26 1.6.4 (2015-10-27)

Bugfixes

- Paths with hyphens in them break the Wheel regular expression. (#145)
- Exception while accessing the `repository` key (sic) when raising a redirect exception. (#146)

10.2.27 1.6.3 (2015-10-05)

Bugfixes

- Fix uploading signatures causing a 500 error after large file support was added. (#137, #140)

10.2.28 1.6.2 (2015-09-28)

Bugfixes

- Upload signatures with packages appropriately (#132)

  As part of the refactor for the 1.6.0 release, we were using the wrong name to find the signature file.

  This also uncovered a bug where if you're using twine in a situation where `*` is not expanded by your shell, we might also miss uploading signatures to PyPI. Both were fixed as part of this.

10.2.29 1.6.1 (2015-09-18)

Bugfixes

- Fix signing support for uploads (#130)

10.2.30 1.6.0 (2015-09-14)

Features

- Allow the user to specify the location of their `.pypirc` (#97)
- Support registering new packages with `twine register` (#8)
- Add the `--skip-existing` flag to `twine upload` to allow users to skip releases that already exist on PyPI. (#115)
- Upload wheels first to PyPI (#106)
- Large file support via the `requests-toolbelt` (#104)

Bugfixes

- Raise an exception on redirects (#92)
- Work around problems with Windows when using `getpass.getpass` (#116)
- Warnings triggered by pkginfo searching for `PKG-INFO` files should no longer be user visible. (#114)

- Provide more helpful messages if `.pypirc` is out of date. (#111)

10.2.31 1.5.0 (2015-03-10)

Features

- Support commands not named "gpg" for signing (#29)

Bugfixes

- Display information about the version of setuptools installed (#85)
- Support deprecated pypirc file format (#61)

Misc

- Add lower-limit to requests dependency

10.2.32 1.4.0 (2014-12-12)

Features

- Switch to a git style dispatching for the commands to enable simpler commands and programmatic invocation. (#6)
- Parse `~/.pypirc` ourselves and use `subprocess` instead of the `distutils.spawn` module. (#13)

Bugfixes

- Expand globs and check for existence of dists to upload (#65)
- Fix issue uploading packages with _s in the name (#47)
- List registered commands in help text (#34)
- Use `pkg_resources` to load registered commands (#32)
- Prevent ResourceWarning from being shown (#28)
- Add support for uploading Windows installers (#26)

10.2.33 1.3.0 (2014-03-31)

Features

- Additional functionality.

10.2.34 1.2.2 (2013-10-03)

Features

- Basic functionality.
- search

Index

P