

---

# **Twine Documentation**

*Release 3.1.2.dev36+gf7402e0*

**Donald Stufft and individual contributors**

**Mar 18, 2020**



---

## Further documentation

---

<b>1 Twine user documentation</b>	<b>3</b>
<b>2 Why Should I Use This?</b>	<b>5</b>
<b>3 Features</b>	<b>7</b>
<b>4 Installation</b>	<b>9</b>
<b>5 Using Twine</b>	<b>11</b>
<b>6 Keyring Support</b>	<b>13</b>
6.1 Disabling Keyring . . . . .	13
<b>7 Options</b>	<b>15</b>
7.1 <code>twine upload</code> . . . . .	15
7.2 <code>twine check</code> . . . . .	16
7.3 <code>twine register</code> . . . . .	16
7.4 Environment Variables . . . . .	17
<b>8 Resources</b>	<b>19</b>
<b>9 Contributing</b>	<b>21</b>
<b>10 Code of Conduct</b>	<b>23</b>
10.1 Contributing . . . . .	23
10.1.1 Getting started . . . . .	23
10.1.2 Architectural overview . . . . .	24
10.1.3 Adding a maintainer . . . . .	25
10.1.4 Making a new release . . . . .	25
10.1.5 Future development . . . . .	25
10.2 Changelog . . . . .	26
<b>Index</b>	<b>31</b>



## Table of Contents

- *Twine user documentation*
- *Why Should I Use This?*
- *Features*
- *Installation*
- *Using Twine*
- *Keyring Support*
  - *Disabling Keyring*
- *Options*
  - *twine upload*
  - *twine check*
  - *twine register*
  - *Environment Variables*
- *Resources*
- *Contributing*
- *Code of Conduct*



# CHAPTER 1

---

## Twine user 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.





---

### 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 3

---

### 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 4

---

### Installation

---

```
$ pip install 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 --repository-url https://test.pypi.org/legacy/ 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](#).





---

## 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](#).

---

### 6.1 Disabling Keyring

In most cases, simply not setting a password in `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.

## 7.1 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).
```

(continues on next page)

(continued from previous page)

```

-i IDENTITY, --identity IDENTITY
                        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.

```

## 7.2 twine check

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

```

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

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

optional arguments:
-h, --help            show this help message and exit

```

## 7.3 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.

```

## 7.4 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, such as a CI/build server, for example.

- `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 8

---

### Resources

---

- [IRC \(#pypa - irc.freenode.net\)](#)
- [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.



Everyone interacting in the `twine` project's codebases, issue trackers, chat rooms, and mailing lists is expected to follow the [PyPA 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 [pypa-dev 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
python3.8 -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 your virtual environment, run:

```
pip install tox
```

## Building the documentation

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

After making docs changes, lint and build the docs locally, using `tox`, before making a pull request. Activate your virtual environment, then, in the root directory, run:

```
tox -e docs
```

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

## Code style

Run `tox -e format` to automatically reformat your changes. Run `tox -e lint` to see any remaining code smells or type errors that need to be fixed manually.

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

## Testing

Twine is tested against Python versions 3.6, 3.7, and 3.8. To run these tests locally, you will need these versions of Python installed on your machine.

Either run `tox` to build against all supported Python versions (if you have them installed) or run `tox -e py{version}` to test against a specific version, e.g., `tox -e py36` or `tox -e py37`.

## 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. Ensure that your name is added 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.

## 10.1.2 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`.

### 10.1.3 Adding a maintainer

A checklist for adding a new maintainer to the project.

1. Add them as a Member in the GitHub repo settings. (This will also give them privileges on the [Travis CI project](#).)
2. Get them Test PyPI and canon PyPI usernames and add them as a Maintainer on [our Test PyPI project](#) and [canon PyPI](#).

### 10.1.4 Making a new release

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

1. Choose a version number, e.g. “1.15.0”
2. Update the changelog:
  1. Add missing changes to `docs/changelog.rst`.
  2. Add a release line at the beginning referencing the release and the date of the release.
  3. Commit, push, ensure Travis build passes.
3. Create a new git tag with `git tag -m tag {number}`.
4. Push the new tag: `git push upstream {number}`.
5. Watch the release [in Travis](#).
6. Send announcement email to [pypa-dev mailing list](#) and celebrate.

### 10.1.5 Future development

See our [open issues](#).

In the future, `pip` and `twine` may merge into a single tool; see [ongoing discussion](#).

## 10.2 Changelog

- #548: Restore `--non-interactive` as a flag not expecting an argument.
- #547: Add support for specifying `--non-interactive` as an environment variable.
- #518: Add Python 3.8 to classifiers.
- #524: 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.
- #489: Add `--non-interactive` flag to abort upload rather than interactively prompt if credentials are missing.
- #336: When a client certificate is indicated, all password processing is disabled.
- #332: More robust handling of server response in `--skip-existing`
- #437: Twine now requires Python 3.6 or later. Use `pip 9` or `pin` to “twine<2” to install twine on older Python versions.
- #491: Require `requests 2.20` or later to avoid reported security vulnerabilities in earlier releases.
- #488: 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.
- #310: Now provide a more meaningful error on redirect during upload.
- #459: Show Warehouse URL after uploading a package
- #456: Better error handling and `gpg2` fallback if `gpg` not available.
- #341: Fail more gracefully when encountering bad metadata
- #416: Add Python 3.7 to classifiers.
- #418: Support `keyring.get_username_and_password`.
- #419: Support `keyring.get_credential`.
- #426: Allow defining an empty username and password in `.pypirc`.
- #427: Add `disable_progress_bar` option to disable `tqdm`.
- #408: Fix `keyring` support.
- #412: Don’t crash if there’s no package description.
- #421: Remove unnecessary usage of `readme_render.markdown`.
- #428: Fix `--skip-existing` for Nexus Repos.
- #432: Use `https` URLs everywhere.
- #435: Specify `python_requires` in `setup.py`
- #436: Use modern Python language features.
- #444: Use `io.StringIO` instead of `StringIO`.
- #441: Only install `pyblake2` if needed.
- #447: Avoid `requests-toolbelt` to 0.9.0 to prevent attempting to use `openssl` when it isn’t available.
- #452: Restore prompts while retaining support for suppressing prompts.
- #439: Refactor `tox env` and `travis config`.

- #404: Fix regression with upload exit code
- #363: Empower `--skip-existing` for Artifactory repositories
- #392: Drop support for Python 3.3
- #395: Add `twine check` command to check long description
- #367: Avoid MD5 when Python is compiled in FIPS mode
- #319: Support Metadata 2.1 ([PEP 566](#)), including Markdown for description fields.
- #320: Remove PyPI as default `register` package index.
- #322: Raise exception if attempting upload to deprecated legacy PyPI URLs.
- #269: Avoid uploading to PyPI when given alternate repository URL, and require `http://` or `https://` in `repository_url`.
- #318: Update PyPI URLs.
- #314: Add new maintainer, release checklists.
- #277: Add instructions on how to use keyring.
- #256: Improve progressbar
- #257: Declare support for Python 3.6
- #303: Revise docs predicting future of `twine`
- #296: Add architecture overview to docs
- #295: Add doc building instructions
- #46: Link to changelog from README
- #304: Reorganize & improve user & developer documentation.
- #265: Fix `--repository[-url]` help text
- #268: Print progress to `stdout`, not `stderr`
- #297: Fix Read the Docs, tox, Travis configuration
- #286: Fix Travis CI and test configuration
- #200: Remove obsolete registration guidance
- #299: Fix changelog formatting
- #298: Fix syntax highlighting in README
- #315: Degrade gracefully when keyring is unavailable
- : Blacklist known bad versions of Requests. See also #253:
- : 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.
- : Fix precedence of `--repository-url` over `--repository`. See also #206:
- : Fix `--skip-existing` when used to upload a package for the first time. See also #220:

- : Twine sends less information about the user’s system in the User-Agent string. See also #229:
- : Twine will use `hashlib.blake2b` on Python 3.6+ instead of using `pyblake2` for Blake2 hashes 256 bit hashes.
- : Twine will now resolve passwords using the `keyring` if available. Module can be required with the `keyring` extra.
- #171: 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.
- #166: Allow the Repository URL to be provided on the command-line (`--repository-url`) or via an environment variable (`TWINE_REPOSITORY_URL`).
- #144: Retrieve configuration from the environment as a default.
  - Repository URL will default to `TWINE_REPOSITORY`
  - Username will default to `TWINE_USERNAME`
  - Password will default to `TWINE_PASSWORD`
- #201: Switch from `upload.pypi.io` to `upload.pypi.org`.
- : Do not generate traffic to Legacy PyPI unless we’re uploading to it or uploading to Warehouse (e.g., `pypi.io`). This avoids the attempt to upload a package to the index if we can find it on Legacy PyPI already.
- : Warn users if they receive a 500 error when uploading to `*pypi.python.org`
- : Stop testing on Python 2.6. 2.6 support will be “best effort” until 2.0.0
- : Generate SHA256 digest for all packages by default.
- : Correct a packaging error.
- #195: 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.
- #189:, #191: Fix issue where we were checking the existence of packages even if the user didn’t specify `--skip-existing`.
- #187: `Clint` was not specified in the wheel metadata as a dependency.
- #177: Switch Twine to upload to `pypi.io` instead of `pypi.python.org`.
- #167: Implement retries when the CDN in front of PyPI gives us a 5xx error.
- #162: Allow `--skip-existing` to work for 409 status codes.
- #152: Add progress bar to uploads.
- #142: 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.
- #186: Allow passwords to have `%s` in them.
- #155: Bump `requests-toolbelt` version to ensure we avoid `ConnectionErrors`
- #146: Exception while accessing the `repository` key (sic) when raising a redirect exception.
- #145: Paths with hyphens in them break the Wheel regular expression.
- #137:, #140: Uploading signatures was broken due to the pull request that added large file support via `requests-toolbelt`. This caused a 500 error on PyPI and prevented package and signature upload in twine 1.6.0
- #132: Upload signatures with packages appropriately



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.

- #130: Fix signing support for uploads
- #8: Support registering new packages with `twine register`
- #115: Add the `--skip-existing` flag to `twine upload` to allow users to skip releases that already exist on PyPI.
- #97: Allow the user to specify the location of their `.pypirc`
- #104: Large file support via the `requests-toolbelt`
- #106: Upload wheels first to PyPI
- #111: Provide more helpful messages if `.pypirc` is out of date.
- #116: Work around problems with Windows when using `getpass.getpass`
- #114: Warnings triggered by `pkginfo` searching for `PKG-INFO` files should no longer be user visible.
- #92: Raise an exception on redirects
- #29: Support commands not named “gpg” for signing
- #61: Support deprecated `pypirc` file format
- #85: Display information about the version of `setuptools` installed
- : Add lower-limit to `requests` dependency
- #6: Switch to a git style dispatching for the commands to enable simpler commands and programmatic invocation.
- #13: Parse `~/ .pypirc` ourselves and use `subprocess` instead of the `distutils.spawn` module.
- #65: Expand globs and check for existence of dists to upload
- #26: Add support for uploading Windows installers
- #47: Fix issue uploading packages with `_s` in the name
- #32: Use `pkg_resources` to load registered commands
- #34: List registered commands in help text
- #28: Prevent `ResourceWarning` from being shown
- : Additional functionality.
- : Basic functionality.
- search



## P

Python Enhancement Proposals

PEP 566, [27](#)