Flit Documentation

Release 1.3

Thomas Kluyver

Feb 07, 2019

Contents

Flit is a simple way to put Python packages and modules on PyPI.

CHAPTER 1

Install

```
$ python3 -m pip install flit
```

Flit requires Python 3 and therefore needs to be installed using the Python 3 version of pip.

Python 2 modules can be distributed using Flit, but need to be importable on Python 3 without errors.

Usage

Say you're writing a module `foobar` — either as a single file `foobar.py`, or as a directory — and you want to distribute it.

1. Make sure that foobar's docstring starts with a one-line summary of what the module is, and that it has a `__version__`:

```
"""An amazing sample package!"""

__version__ = '0.1'
```

2. Install flit if you don't already have it:

```
python3 -m pip install flit
```

3. Run `flit init` to create a `pyproject.toml` file. It will look something like this:

```
[build-system]
requires = ["flit"]
build-backend = "flit.buildapi"

[tool.flit.metadata]
module = "foobar"
author = "Sir Robin"
author-email = "robin@camelot.uk"
home-page = "https://github.com/sirrobin/foobar"
```

You can edit this file to add other metadata, for example to set up command line scripts. See the pyproject.toml page of the documentation.

If you have already got a `flit.ini` file to use with older versions of Flit, it will still work for now, but you should convert it to `pyproject.toml` when convenient.

4. Run this command to upload your code to PyPI:

```
flit publish
```

To install a package locally for development, run:

```
flit install [--symlink] [--python path/to/python]
```

Flit packages a single importable module or package at a time, using the import name as the name on PyPI. All subpackages and data files within a package are included automatically.

Development

To install the development version of Flit from Github:

```
git clone https://github.com/takluyver/flit.git
cd flit
python3 -m pip install docutils requests pytoml
python3 -m flit install
```

You may want to use the `--symlink` or `--pth-file` options so you can test changes without reinstalling it.

To run the tests, run `pytest.`

Documentation contents

4.1 The pyproject.toml config file

This file lives next to the module or package.

Note: Older version of Flit (up to 0.11) used a flit.ini file for similar information. Flit can still read these files for now, but you should switch to `pyproject.toml` soon.

Run `python3 -m flit.tomlify` to convert a `flit.ini` file to `pyproject.toml`.

4.1.1 Build system section

This tells tools like pip to build your project with flit. It's a standard defined by PEP 517. For any project using Flit, it will look like this:

```
[build-system]
requires = ["flit"]
build-backend = "flit.buildapi"
```

4.1.2 Metadata section

This section is called `[tool.flit.metadata]` in the file. There are three required fields:

module The name of the module/package, as you'd use in an import statement.

author Your name

author-email Your email address

e.g. for flit itself

```
[tool.flit.metadata]
module = "flit"
author = "Thomas Kluyver"
author-email = "thomas@kluyver.me.uk"
```

Changed in version 1.1: `home-page` was previously required.

The remaining fields are optional:

home-page A URL for the project, such as its Github repository.

requires A list of other packages from PyPI that this package needs. Each package may be followed by a version specifier like (`>=4.1`) or `>=4.1`, and/or an environment marker after a semicolon. For example:

```
requires = [
    "requests >=2.6",
    "configparser; python_version == '2.7'",
]
```

requires-extra Lists of packages needed for every optional feature. The requirements are specified in the same format as for `requires`. The requirements of the two reserved extras `test` and `doc` as well as the extra `dev` are installed by `flit install`. For example:

```
[tool.flit.metadata.requires-extra]
test = [
    "pytest >=2.7.3",
    "pytest-cov",
]
doc = ["sphinx"]
```

New in version 1.1.

description-file A path (relative to the .toml file) to a file containing a longer description of your package to show on PyPI. This should be written in reStructuredText, Markdown or plain text, and the filename should have the appropriate extension (`.rst`, `.md` or `.txt`).

classifiers A list of Trove classifiers.

requires-python A version specifier for the versions of Python this requires, e.g. `~=3.3` or `>=3.3,<4` which are equivalents.

dist-name If you want your package's name on PyPI to be different from the importable module name, set this to the PyPI name.

keywords Space separated list of words to help with searching for your package.

license The name of a license, if you're using one for which there isn't a Trove classifier. It's recommended to use Trove classifiers instead of this in most cases.

maintainer, maintainer-email Like author, for if you've taken over a project from someone else.

Here's the full metadata section from flit itself:

```
[tool.flit.metadata]
module="flit"
author="Thomas Kluyver"
author-email="thomas@kluyver.me.uk"
home-page="https://github.com/takluyver/flit"
requires=[
    "requests",
```

(continues on next page)

```
    "docutils",
    "requests_download",
    "pytoml",
]
requires-python="3"
description-file="README.rst"
classifiers=[
    "Intended Audience :: Developers",
    "License :: OSI Approved :: BSD License",
    "Programming Language :: Python :: 3",
    "Topic :: Software Development :: Libraries :: Python Modules",
]
```

URLs subsection

Your project's page on pypi.org can show a number of links, in addition to the required `home-page` URL described above. You can point people to documentation or a bug tracker, for example.

This section is called `[tool.flit.metadata.urls]` in the file. You can use any names inside it. Here it is for flit:

```
[tool.flit.metadata.urls]
Documentation = "https://flit.readthedocs.io/en/latest/"
```

New in version 1.0.

4.1.3 Scripts section

This section is called `[tool.flit.scripts]` in the file. Each key and value describes a shell command to be installed along with your package. These work like setuptools 'entry points'. Here's the section for flit:

```
[tool.flit.scripts]
flit = "flit:main"
```

This will create a `flit` command, which will call the function `main()` imported from `flit`.

4.1.4 Entry points sections

You can declare entry points using sections named `[tool.flit.entrypoints.`*`groupname`*`]`. E.g. to provide a pygments lexer from your package:

```
[tool.flit.entrypoints."pygments.lexers"]
dogelang = "dogelang.lexer:DogeLexer"
```

In each `package:name` value, the part before the colon should be an importable module name, and the latter part should be the name of an object accessible within that module. The details of what object to expose depend on the application you're extending.

4.2 Flit command line interface

All operations use the `flit` command, followed by one of a number of subcommands.

4.2.1 Common options

-f `<path>`, **--ini-file** `<path>`
　　Path to a config file specifying the module to build. The default is `pyproject.toml` or `flit.ini`

--repository `<repository>`
　　Name of a repository to upload packages to. Should match a section in `~/.pypirc`. The default is `pypi`. See *Controlling package uploads*.

--version
　　Show the version of Flit in use.

--help
　　Show help on the command-line interface.

--debug
　　Show more detailed logs about what flit is doing.

4.2.2 `flit build`

Build a wheel and an sdist (tarball) from the package.

--format `<format>`
　　Limit to building either `wheel` or `sdist`.

4.2.3 `flit publish`

Build a wheel and an sdist (tarball) from the package, and upload them to PyPI or another repository.

--format `<format>`
　　Limit to publishing either `wheel` or `sdist`. You should normally publish the two formats together.

See also:

Controlling package uploads

4.2.4 `flit install`

Install the package on your system.

-s, **--symlink**
　　Symlink the module into site-packages rather than copying it, so that you can test changes without reinstalling the module.

--pth-file
　　Create a `.pth` file in site-packages rather than copying the module, so you can test changes without reinstalling. This is a less elegant alternative to `--symlink`, but it works on Windows, which typically doesn't allow symlinks.

--deps `<dependency option>`
　　Which dependencies to install. One of `all`, `production`, `develop`, or `none`. `all` and `develop` install the extras `test`, `doc`, and `dev`. Default `all`.

--extras `<extra[,extra,...]>`
　　Which named extra features to install dependencies for. Specify `all` to install all optional dependencies, or a comma-separated list of extras. Default depends on `--deps`.

--user
> Do a user-local installation. This is the default if flit is not in a virtualenv or conda env (if the environment's library directory is read-only and `site.ENABLE_USER_SITE` is true).

--env
> Install into the environment. This is the default in a virtualenv or conda env (if the environment's library directory is writable or `site.ENABLE_USER_SITE` is false).

--python `<path to python>`
> Install for another Python, identified by the path of the python executable. The default is to install the module for the copy of Python that is running Flit. Using this option, you can install a module for Python 2, for instance.

Note: Flit calls pip to do the installation. You can set any of pip's options using its environment variables.

When you use the *--symlink* or *--pth-file* options, pip is used to install dependencies. Otherwise, Flit builds a wheel and then calls pip to install that.

4.2.5 `flit init`

Create a new `pyproject.toml` config file by prompting for information about the module in the current directory.

4.2.6 Environment variables

FLIT_NO_NETWORK
> New in version 0.10.
>
> Setting this to any non-empty value will stop flit from making network connections (unless you explicitly ask to upload a package). This is intended for downstream packagers, so if you use this, it's up to you to ensure any necessary dependencies are installed.

FLIT_ROOT_INSTALL
> By default, `flit install` will fail when run as root on POSIX systems, because installing Python modules systemwide is not recommended. Setting this to any non-empty value allows installation as root. It has no effect on Windows.

FLIT_USERNAME
FLIT_PASSWORD
FLIT_INDEX_URL
> New in version 0.11.
>
> Set a username, password, and index URL for uploading packages. See *uploading packages with environment variables* for more information.

FLIT_ALLOW_INVALID
> New in version 0.13.
>
> Setting this to any non-empty value tells Flit to continue if it detects invalid metadata, instead of failing with an error. Problems will still be reported in the logs, but won't cause Flit to stop.
>
> If the metadata is invalid, uploading the package to PyPI may fail. This environment variable provides an escape hatch in case Flit incorrectly rejects your valid metadata. If you need to use it and you believe your metadata is valid, please open an issue.

SOURCE_DATE_EPOCH
> To make reproducible builds, set this to a timestamp as a number of seconds since the start of the year 1970 in UTC, and document the value you used. On Unix systems, you can get a value for the current time by running:

```
date +%s
```

See also:

The SOURCE_DATE_EPOCH specification

4.3 Controlling package uploads

The command `flit publish` will upload your package to a package index server. The default settings let you upload to PyPI, the default Python Package Index, with a single user account.

If you want to upload to other servers, or with more than one user account, or upload packages from a continuous integration job, you can configure Flit in two main ways:

4.3.1 Using .pypirc

You can create or edit a config file in your home directory, `~/.pypirc`. This is also used by other Python tools such as twine.

For instance, to upload a package to the Test PyPI server instead of the normal PyPI, use a config file looking like this:

```
[distutils]
index-servers =
   pypi
   testpypi

[pypi]
repository = https://upload.pypi.org/legacy/
username = sirrobin  # Replace with your PyPI username

[testpypi]
repository = https://test.pypi.org/legacy/
username = sirrobin  # Replace with your TestPyPI username
```

You can select an index server from this config file with the `--repository` option:

```
flit --repository testpypi publish
```

If you don't use this option, Flit will use the server called `pypi` in the config file. If that doesn't exist, it uploads to PyPI at `https://upload.pypi.org/legacy/` by default.

If you publish a package and you don't have a `.pypirc` file, Flit will create it to store your username.

Flit tries to store your password securely using the keyring library. If keyring is not installed, Flit will ask for your password for each upload. Alternatively, you can also manually add your password to the `.pypirc` file (`password = ...`)

4.3.2 Using environment variables

You can specify a server to upload to with *FLIT_INDEX_URL*, and pass credentials with *FLIT_USERNAME* and *FLIT_PASSWORD*. Environment variables take precedence over the config file, except if you use the `--repository` option to explicitly pick a server from the config file.

This can make it easier to automate uploads, for example to release packages from a continuous integration job.

> **Warning:** Storing a password in an environment variable is convenient, but it's easy to accidentally leak it. Look out for scripts that helpfully print all environment variables for debugging, and remember that other scripts and libraries you run in that environment have access to your password.

4.4 Reproducible builds

New in version 0.8.

Wheels built by flit are reproducible: if you build from the same source code, you should be able to make wheels that are exactly identical, byte for byte. This is useful for verifying software. For more details, see reproducible-builds.org.

There is a caveat, however: wheels (which are zip files) include the modification timestamp from each file. This will probably be different on each computer, because it indicates when your local copy of the file was written, not when it was changed in version control. These timestamps can be overridden by the environment variable *SOURCE_DATE_EPOCH*.

```
SOURCE_DATE_EPOCH=$(date +%s)
flit publish
# Record the value of SOURCE_DATE_EPOCH in release notes for reproduction
```

Changed in version 0.12: Normalising permission bits

Flit normalises the permission bits of files copied into a wheel to either 755 (executable) or 644. This means that a file is readable by all users and writable only by the user who owns it.

The most popular version control systems only track the executable bit, so checking out the same repository on systems with different umasks (e.g. Debian and Fedora) produces files with different permissions. With Flit 0.11 and earlier, this difference would produce non-identical wheels.

4.5 Release history

4.5.1 Version 1.3

- Fix for building sdists from a subdirectory in a Mercurial repository (PR #233).
- Fix for getting the docstring and version from modules defining their encoding (PR #239).
- Fix for installing packages with `flit installfrom` (PR #221).
- Packages with requirements no longer get a spurious `Provides-Extra: .none` metadata entry (#228).
- Better check of whether `python-requires` includes any Python 2 version (PR #232).
- Better check of home page URLs in `flit init` (PR #230).
- Better error message when the description file is not found (PR #234).
- Updated a help message to refer to `pyproject.toml` (PR #240).
- Improve tests of `flit init` (PR #229).

4.5.2 Version 1.2.1

- Fix for installing packages with `flit install`.

- Make `requests_download` an extra dependency, to avoid a circular build dependency. To use `flit installfrom`, you can install with `pip install flit[installfrom]`. Note that the `installfrom` subcommand is deprecated, as it will soon be possible to use pip to install Flit projects directly from a VCS URL.

4.5.3 Version 1.2

- Fixes for packages specifying `requires-extra`: sdists should now work, and environment markers can be used together with `requires-extra`.
- Fix running `flit installfrom` without a config file present in the working directory.
- The error message for a missing or empty docstring tells you what file the docstring should be in.
- Improvements to documentation on version selectors for requirements.

4.5.4 Version 1.1

- Packages can now have 'extras', specified as `requires-extra` in the *pyproject.toml file*. These are additional dependencies for optional features.
- The `home-page` metadata field is no longer required.
- Additional project URLs are now validated.
- `flit -V` is now equivalent to `flit --version`.
- Various improvements to documentation.

4.5.5 Version 1.0

- The description file may now be written in reStructuredText, Markdown or plain text. The file extension should indicate which of these formats it is (`.rst`, `.md` or `.txt`). Previously, only reStructuredText was officially supported.
- Multiple links (e.g. documentation, bug tracker) can now be specified in a new *[tool.flit.metadata.urls] section* of `pyproject.toml`.
- Dependencies are now correctly installed to the target Python when you use the `--symlink` or `--pth-file` options.
- Dependencies are only installed to the Python where Flit is running if it fails to get the docstring and version number without them.
- The commands deprecated in 0.13—`flit wheel`, `flit sdist` and `flit register`—have been removed.

Although version 1.0 sounds like a milestone, there's nothing that makes this release especially significant. It doesn't represent a step change in stability or completeness. Flit has been gradually maturing for some time, and I chose this point to end the series of 0.x version numbers.

4.5.6 Version 0.13

- Better validation of several metadata fields (`dist-name`, `requires`, `requires-python`, `home-page`), and of the version number.
- New *FLIT_ALLOW_INVALID* environment variable to ignore validation failures in case they go wrong.

- The list of valid classifiers is now fetched from Warehouse (https://pypi.org), rather than the older https://pypi.python.org site.

- Deprecated `flit wheel` and `flit sdist` subcommands: use *flit build*.

- Deprecated `flit register`: you can no longer register a package separately from uploading it.

4.5.7 Version 0.12.3

- Fix building and installing packages with a - in the distribution name.

- Fix numbering in README.

4.5.8 Version 0.12.2

- New tool to convert `flit.ini` to `pyproject.toml`:

```
python3 -m flit.tomlify
```

- Use the PAX tar format for sdists, as specified by PEP 517.

4.5.9 Version 0.12.1

- Restore dependency on `zipfile36` backport package.

- Add some missing options to documentation of `flit install` subcommand.

- Rearrange environment variables in the docs.

4.5.10 Version 0.12

- Switch the config to `pyproject.toml` by default instead of `flit.ini`, and implement the PEP 517 API.

- A new option `--pth-file` allows for development installation on Windows (where `--symlink` usually won't work).

- Normalise file permissions in the zip file, making builds more reproducible across different systems.

- Sdists (.tar.gz packages) can now also be reproducibly built by setting *SOURCE_DATE_EPOCH*.

- For most modules, Flit can now extract the version number and docstring without importing it. It will still fall back to importing where getting these from the AST fails.

- `flit build` will build the wheel from the sdist, helping to ensure that files aren't left out of the sdist.

- All list fields in the INI file now ignore blank lines (`requires`, `dev-requires`, `classifiers`).

- Fix the path separator in the `RECORD` file of a wheel built on Windows.

- Some minor fixes to building reproducible wheels.

- If building a wheel fails, the temporary file created will be cleaned up.

- Various improvements to docs and README.

4.5.11 Version 0.11.4

- Explicitly open various files as UTF-8, rather than relying on locale encoding.

- Link to docs from README.

- Better test coverage, and a few minor fixes for problems revealed by tests.

4.5.12 Version 0.11.3

- Fixed a bug causing failed uploads when the password is entered in the terminal.

4.5.13 Version 0.11.2

- A couple of behaviour changes when uploading to warehouse.

4.5.14 Version 0.11.1

- Fixed a bug when you use flit to build an sdist from a subdirectory inside a VCS checkout. The VCS is now correctly detected.

- Fix the rst checker for newer versions of docutils, by upgrading the bundled copy of readme_renderer.

4.5.15 Version 0.11

- Flit can now build sdists (tarballs) and upload them to PyPI, if your code is in a git or mercurial repository. There are new commands:

  - `flit build` builds both a wheel and an sdist.

  - `flit publish` builds and uploads a wheel and an sdist.

- Smarter ways of getting the information needed for upload:

  - If you have the keyring package installed, flit can use it to store your password, rather than keeping it in plain text in `~/.pypirc`.

  - If `~/.pypirc` does not already exist, and you are prompted for your username, flit will write it into that file.

  - You can provide the information as environment variables: `FLIT_USERNAME`, `FLIT_PASSWORD` and `FLIT_INDEX_URL`. Use this to upload packages from a CI service, for instance.

- Include 'LICENSE' or 'COPYING' files in wheels.

- Fix for `flit install --symlink` inside a virtualenv.

4.5.16 Version 0.10

- Downstream packagers can use the `FLIT_NO_NETWORK` environment variable to stop flit downloading data from the network.

4.5.17 Version 0.9

- `flit install` and `flit installfrom` now take an optional `--python` argument, with the path to the Python executable you want to install it for. Using this, you can install modules to Python 2.

- Installing a module normally (without `--symlink`) builds a wheel and uses pip to install it, which should work better in some corner cases.

4.5.18 Version 0.8

- A new `flit installfrom` subcommand to install a project from a source archive, such as from Github.

- *Reproducible builds* - you can produce byte-for-byte identical wheels.

- A warning for non-canonical version numbers according to PEP 440.

- Fix for installing projects on Windows.

- Better error message when module docstring is only whitespace.

4.5.19 Version 0.7

- A new `dev-requires` field in the config file for development requirements, used when doing `flit install`.

- Added a `--deps` option for `flit install` to control which dependencies are installed.

- Flit can now be invoked with `python -m flit`.

4.5.20 Version 0.6

- `flit install` now ensures requirements specified in `flit.ini` are installed, using pip.

- If you specify a description file, flit now warns you if it's not valid reStructuredText (since invalid reStructuredText is treated as plain text on PyPI).

- Improved the error message for mis-spelled keys in `flit.ini`.

4.5.21 Version 0.5

- A new `flit init` command to quickly define the essential basic metadata for a package.

- Support for entrypoints.

- A new `flit register` command to register a package without uploading it, for when you want to claim a name before you're ready to release.

- Added a `--repository` option for specifying an alternative PyPI instance.

- Added a `--debug` flag to show debug-level log messages.

- Better error messages when the module docstring or `__version__` is missing.

4.5.22 Version 0.4

- Users can now specify `dist-name` in the config file if they need to use different names on PyPI and for imports.

- Classifiers are now checked against a locally cached list of valid classifiers.

- Packages can be locally installed into environments for development.

- Local installation now creates a PEP 376 `.dist-info` folder instead of `.egg-info`.

CHAPTER 5

Indices and tables

- genindex
- search

Symbols

E

F

S