Managing Dependencies With Poetry

Christian Külker

2022-05-25

Contents

As many other fancy tools poetry is used to circumvent the well maintained package manager of your distribution and this is called “Dependency Management for Python” or as python-poetry.org describes: “Python packaging and dependency management made easy”.

Unlike pipenv, poetry is not in Debian, so it has to be downloaded from an unsupervised location of the internet: github or pypi.

1 Installation

We leave out the insecure curl method. In the github README you can read the warning “Be aware, however, that it will also install poetry’s dependencies which might cause conflicts.”How can that be that a dependency manager needs to be afraid of conflicting dependencies? Isn’t that the reason to use it in the first place. For me this reads more like prose not poetry. Welcome to the new era of package “managers”!

```
$ python3 -m pip install --user poetry
[...]
Successfully installed attrs-19.3.0 cachecontrol-0.12.6 cachy-0.3.0 \
certifi-2020.4.5.1 cffi-1.14.0 chardet-3.0.4 cleo-0.7.6 clikit-0.4.3 \
cryptography-2.9.2 html5lib-1.0.1 idna-2.9 importlib-metadata-1.1.3 \
jeepney-0.4.3 jsonschema-3.2.0 keyring-20.0.1 lockfile-0.12.2 \
msgpack-1.0.0 pastel-0.2.0 pexpect-4.8.0 pkginfo-1.5.0.1 \
poetry-1.0.5 ptyprocess-0.6.0 pycparser-2.20 pylev-1.3.0
```

2 One File To Bind Them All

The tool poetry handle dependency installation, building and packaging. It only needs one file: the PEP518 `pyproject.toml` and replaces `setup.py`, `requirements.txt`, `setup.cfg`, `MANIFEST.in` and `Pipfile`. Note, that not all files are needed by other dependency managers.

```
[tool.poetry]
name = "example-pkg-ckuelker"
version = "0.0.1"
description = "A small example package"
license = "GPLv3"
authors = [
    "Christian Külker <test-pypi-org@c8i.org>",
]
readme = 'README.md'  # Markdown files are supported
repository = "https://github.com/ckuelker/python-packaging-tutorial-
example-package"
homepage  =  "https://github.com/ckuelker/python-packaging-tutorial-
example-package"
keywords = ['packaging', 'tutorial']

[tool.poetry.dependencies]
python = "~2.7 || ^3.2"  # Compatible python versions must be declared here
toml = "^0.9"
### Dependencies with extras
requests = { version = "^2.13", extras = [ "security" ] }
### Python specific dependencies with prereleases allowed
pathlib2 = { version = "^2.2", python = "~2.7", allow-prereleases = true }
### Very "secure" Git dependencies
cleo = { git = "https://github.com/sdispater/cleo.git", branch = "master" }
```

```
### Optional dependencies (extras)
pendulum = { version = "^1.4", optional = true }

[tool.poetry.dev-dependencies]
pytest = "^3.0"
pytest-cov = "^2.4"

[tool.poetry.scripts]
my-script = 'my_package:main'
```

3 Installing the dependencies

```
poetry install
Updating dependencies
Resolving dependencies... (21.7s)

Writing lock file


Package operations: 9 installs, 5 updates, 0 removals

  - Updating zipp (3.1.0 -> 1.2.0)
  - Installing atomicwrites (1.4.0)
  - Updating cryptography (2.9.2 -> 2.8)
  - Installing more-itertools (5.0.0)
  - Installing pluggy (0.13.1)
  - Installing py (1.8.1)
  - Installing coverage (4.5.4)
  - Updating idna (2.9 -> 2.8)
  - Installing pyopenssl (19.1.0)
  - Installing pytest (3.10.1)
  - Updating urllib3 (1.25.9 -> 1.24.3)
  - Installing pytest-cov (2.8.1)
  - Updating requests (2.23.0 -> 2.21.0)
  - Installing toml (0.9.6)
```

This creates a big `poetry.lock` file. Unlike working with `setup.py` working with test.pypi.org is not as straight forward.

```
$ poetry build
Building example-pkg-ckuelker (0.0.1)

[ModuleOrPackageNotFound]
No file/folder found for package example-pkg-ckuelker
```

Removing the name from the username creates the package.

```
$ poetry build
Building example-pkg (0.0.1)
 - Building sdist
 - Built example-pkg-0.0.1.tar.gz

 - Building wheel
 - Built example_pkg-0.0.1-py2.py3-none-any.whl
$ tree
dist
├── example_pkg-0.0.1-py2.py3-none-any.whl
├── example-pkg-0.0.1.tar.gz
```

However this do not conform to the test.pypi.org format:

```
$ tree
dist
├── example_pkg_ckuelker-0.0.1-py3-none-any.whl
└── example-pkg-ckuelker-0.0.1.tar.gz
```

Renaming the directory from `example_pkg` to `example_pkg_ckuelker` did the trick:

```
$ poetry build
Building example-pkg_ckuelker (0.0.1)
 - Building sdist
 - Built example-pkg_ckuelker-0.0.1.tar.gz

 - Building wheel
 - Built example_pkg_ckuelker-0.0.1-py2.py3-none-any.whl
$ tree dist
dist
├── example_pkg_ckuelker-0.0.1-py2.py3-none-any.whl
└── example-pkg_ckuelker-0.0.1.tar.gz
```

If your project has a unique name, testing with poetry works, if not renaming is the way to go: this disqualifies poetry to use in package tutorials.

An alternative is to create a symbolic link.

```
ln -s example_pkg example_pkg_YOUR_USERNAME
```

On how to install the package and on how to upload this to test.pypi.org see packaging-python-projects.

4 Considerations

Reading the reasoning behind poetry gives the impression of a **not invented here** project: about pipenv "I do not like the CLI it provides, or some of the decisions made, and I think we can make a better and more intuitive one." Intuitiveness is the best for a software one writes by it oneself. Writing a tool because dependency management is "convoluted" and "hard to understand" for newcomers is a non argument. Dependencies management is always difficult to understand. The project claims "[…] there is no reliable tool to properly resolve dependencies in Python". I doubt it and my answer: there is Debian. Usually what I expect is: Feature X is missing, that is why I wrote software Y. None of this seem to be a reason for this project.

However, the project is correct when pointing on problems of `pipenv installing oslo.utils==1.4.0`. Meanwhile also poetry has its problems (and has bad error messages).

```
poetry add oslo.utils=1.4.0

[InvalidCharInStringError]
Invalid character '\n' in string at line 11 col 81
```

Which is not related to `oslo`, just a " (quote character) was missing in line 11 of `pyproject.toml`. Who would have thought that? Actually it added `oslo` fine with the " (quote character) fixed:

```
poetry add oslo.utils=1.4.0

Updating dependencies
Resolving dependencies... (14.9s)

Writing lock file


Package operations: 0 installs, 1 update, 0 removals

  - Updating oslo.i18n (4.0.1 -> 2.1.0)
```

Not sure if this is an update, though. I would have expected `2.1.0 -> 4.0.1`. Looks more like a downgrade …

But the colorful console characters are looking very cheerfully.

5 History

Version	Date	Notes
0.1.3	2022-05-25	Change comments, replace shell with bash
0.1.2	2022-05-09	Fix missing quotes in toml section
0.1.1	2020-09-05	Fix heading levels, fix and add more links
0.1.0	2020-05-18	Initial release

6 Disclaimer of Warranty

7 Limitation of Liability