Poetry 101

Basics and stumbling points

YOSHIHIRO FUKUHARA

cvpaper.challenge
May. 30th, 2022

Content

- What is Poetry?
- How Poetry work?
- Frequent commands
- Dependency specification
- Frequently faced error

What is Poetry?

A tool for dependency management and packaging in Python.

Dependency Management for Python

v1.1.13 6.9M/month

Stars 20k

FEATURE

- Dependency management by exhaustive resolver
- Environment isolation by virtualenvs
- Easily buid, package and publish projects to PyPI

How Poetry work? `StumblingPoints`

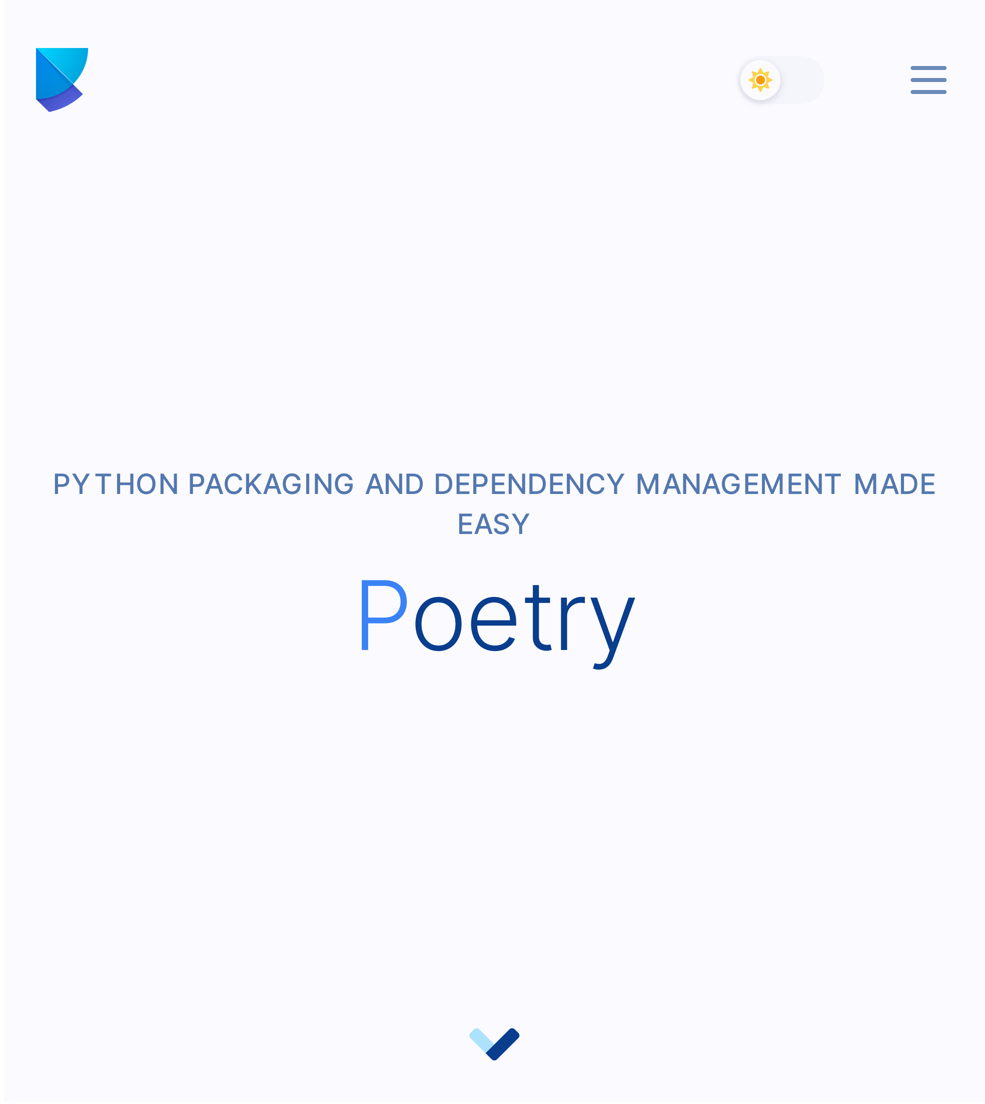
- Two important files:
  - `pyproject.toml`: (constraint) A file that is modified by the user; e.g. `poetry add` command. It contains constraints on the version of the package **to be installed.**
  - `poetry.lock`: (state) A file that is automatically modified by Poetry. It contains info about the packages **actually installed** and their dependencies, based on the constraints in `pyproject.toml`.
- When Poetry installs or updates a package, it checks whether a version that satisfies both 1. and 2. below exists, and executes it if it is found. (In this slide, 1. and 2. are combined and referred to as **updateability requirements.**)

  1. **All packages must sutisfy the version constraints described in `pyproject.toml`.**
  2. **No conflicts with already installed packages and their dependent packages.**

- By including the `poetry.lock` file and sharing it on github etc., you can share the exactly same Python package environment among your team.

Frequent commands

- example commands:
  - install
  - add
  - update
  - remove
  - run
  - show
- For detail please check offical docs.


PYTHON PACKAGING AND DEPENDENCY MANAGEMENT MADE EASY

Poetry

Frequent commands

install

```
$ poetry install

# install packages except dev dependencies.
$ poetry install --no-dev
```

- Refer to the version constraints in `pyproject.toml`, search for the version of each package that satisfies the updatable requirement, and install if it found. At the same time, create `poetry.lock` and write the information of the installed packages.
- If `poetry.lock` already exists, install the exact same version of the package as described in `poetry.lock`.
- If a config `virtualenvs.create` is True, create a virtual environment and packages are installed in it.

Frequent commands

add

`StumblingPoints`

```
# Check if the latest version of numpy satisfies the
# updatable requirements. If so, then install it.
$ poetry add numpy

# Search for versions of numpy less than x.y that meets
# the updatable requirements. If it found, install it.
$ poetry add "numpy<x.y"

# (Caret requirements) Search for versions in the range
# of x.y.z or greater, and less than x+1.0.0.
$ poetry add "numpy^x.y.z"

# Attempt to install the master branch of pytorch.
$ poetry add git@github.com:pytorch/pytorch.git#master

# Attempt to install local directory /my-package.
$ poetry add ./my-package/
```

- Add a package if the version that satisfies the updatable requirements is found. (If not, a `` `SolverProblemError` `` is raised.)
- If it is a package, you can also install specific branches of github or local directories/files.
- There are several unique notations for version specification (explain later).
- While useful, it is also a command that can be easily stumbled over due to errors (explain how to deal with errors later).

Frequent commands

update

```
# Update all packages that satisfies the updatable
# requirements.
$ poetry update

# It can also be used for specific packages only.
$ poetry update numpy
```

- Update the package if it satisfies the updatable requirements.
- A list of packages that can be updated is able to be found by `` `poetry show --latest` `` which is explained later.
- When you want to update exceeding the version constraints listed in `` `pyproject.toml` ``, need to use `` `poetry add` `` to add the package again.

Frequent commands

remove

```
# uninstall numpy.
$ poetry remove numpy
```

- When `` `poetry remove` `` is executed, the version constraint information of the target package in `` `pyproject.toml` `` is removed.
- If no other package depends on the target package, the package will be uninstalled.

Frequent commands

run

```
# Run Python3 in the virtual environment created by Poetry.
$ poetry run python3

# Apply black to the src directory in the virtual
# environment created by Poetry.
$ poetry run black src
```

- Running commands in the Poetry virtual environment. To use packages installed by Poetry, you need to execute the commands in the Poetry virtual environment.
- If you have installed a package but it is not found, you might forgot to use this `` `poetry run` ``.
- If you don't want to use `` `poetry run` `` each time, you can use the `` `poetry shell` `` command to start a new shell in a virtual environment.

Frequent commands

show

```
# Shows the list of currently installed packages.
$ poetry show

# Shows package dependencies as a tree.
$ poetry show --tree

# Shows the latest version of the package.
$ poetry show --latest
```

- Shows various information about the packages installed by Poetry.
- In particular, `` `poetry show --latest` `` is useful in combination with `` `poetry update` ``.

Off-topic

Python module / package / library

- Python module means a `` `.py` `` file.
- Python package is a way of structuring a module and means a directory containing `` `__init__.py` `` and `` `.py` `` files. A package may contain subordinate packages within it.
- The definition of "library" is not mentioned in the official Python documentation, but it often refers to a package or a set of packages published to PyPI etc.

Off-topic

Semantic versioning

- Specify the software version in the form of `` `x.y.z` ``.
- `` `x` `` is called the **major version** and is incremented when the API changes incompatibly.
- `` `y` `` is called the **minor version** and is incremented when backward-compatible functionality is added.
- `` `z` `` is called the **patch version**, and is incremented when a bug fix with backward compatibility is made.

Dependency specification

Caret requirements `StumblingPoints`

- Specify version using `` `^` ``.
- Allows a range of non-zero most-left digit is not change.

Requirement	Versions allowed
`` `^1.2.3` ``	`` `>=1.2.3,<2.0.0` ``
`` `^1.2` ``	`` `>=1.2.0,<2.0.0` ``
`` `^1` ``	`` `>=1.0.0,<2.0.0` ``
`` `^0.2.3` ``	`` `>=0.2.3,<0.3.0` ``
`` `^0.0.3` ``	`` `>=0.0.3,<0.0.4` ``

Dependency specification

Tilde requirements

- Specify version using `` `~` ``.
- The meaning differs depending on its format:
    - When in `` `~x.y.z` `` or `` `~x.y` `` format, allow a range of patch version changes.
    - When in `` `~x` `` format, allow a range of minor version changes.

Requirement	Versions allowed
`` `~1.2.3` ``	`` `>=1.2.3,<1.3.0` ``
`` `~1.2` ``	`` `>=1.2.0,<1.3.0` ``
`` `~1` ``	`` `>=1.0.0,<2.0.0` ``

Frequently faced error

SolverProblemError

StumblingPoints

```
# Attempt to install two packages related to AWS
$ poetry add "boto3==1.16.43"
$ poetry add "s3fs^2022.5.0"

Updating dependencies
Resolving dependencies... (0.4s)

  SolverProblemError

  (途中略)
  Thus, s3fs (>=2022.5.0,<2023.0.0) requires botocore (>=1.2
  And because boto3 (1.16.43) depends on botocore (>=1.19.43
  So, because ascender depends on both boto3 (1.16.43) and s
```

- `` `s3fs` `` depends on `` `botocore(>=1.24.21,<1.24.22)` `` and `` `boto3` `` depends on `` `botocore(>=1.19.43,<1.20.0)` `` respectively. This raises a `` `SolverProblemError` `` because the updateability requirement 2. is not satisfied.
- To install both `` `s3fs` `` and `` `boto3` ``, you need to adjust version constraints to avoid `` `botocore` `` conflicts.

Frequently faced error

SolverProblemError `StumblingPoints`

```
# Attempt to install two packages related to AWS
$ poetry add "boto3==1.16.43"
$ poetry add "s3fs<=2022.5.0" # relax the constraint of s3fs

Updating dependencies
Resolving dependencies... (8.4s)

Writing lock file

Package operations: 2 installs, 0 updates, 0 removals

  • Installing fsspec (2022.5.0)
  • Installing s3fs (0.4.2)
```

- For example, by relaxing the version constraint of `` `s3fs` ``, Poetry will search for a version of `` `s3fs` `` that does not conflict with the version of `` `botocore` `` on which `` `boto3` `` depends.
- If you loosen the `` `s3fs` `` version constraint as above and still cannot find a version that does not conflict, you will get a `` `SolverProblemError` `` as well. In such cases, it is necessary to consider relaxing the `` `boto3` `` version constraint as well.

Yoshihiro Fukuhara

ML / MLOps engineer.

HQ member and XCCV group lead of cvpaper.challenge.

gatheluck

gatheluck

gatheluck.net

cvpaper.challenge

Visit our page

END