Monas

Release 0.1.0

Frost Ming

Apr 17, 2022

CONTENTS

1 About this project **3**
- 1.1 Quick Start . . . 3
- 1.2 Integration . . . 4
- 1.3 `monas` . . . 5
- 1.4 Changelog . . . 9
- 1.5 Contributing . . . 10

Index **13**

Python monorepo made easy.

[WIP] This project still in a rapid development and the behaviors may change.

Example Repository

CHAPTER

ONE

ABOUT THIS PROJECT

Monas is a tool to manage multiple Python projects in a single repository, or the so called "Monorepo". It is mainly inspired by Lerna. In a monorepo, some dependencies are shared across packages while others are different. When you change the code of one of these shared dependencies, you may want to run the test suite across all dependant packages. Monas makes the workflow easier.

1.1 Quick Start

1.1.1 Install Monas

Monas requires Python >=3.7.

It is recommended to install with `pipx`, if `pipx` haven't been installed yet, refer to the pipx's docs

```
pipx install monas
```

Alternatively, install with `pip` to the user site:

```
python -m pip install --user monas
```

1.1.2 Create a monorepo

Monas is integrated with Git. You need to install it if it isn't on your system.

```
git init mono-project
cd mono-project
monas init
```

1.1.3 Add a subpackage

```
monas new foo
```

Answer a few questions and the subpackage `foo` will be created under `packages/` directory.

See what packages are added:

```
monas list
```

1.1.4 Install all packages and dependencies

```
monas install
```

Monas will create a virtualenv under each subpackage and install all dependencies into it. The subpackage itself and other subpackages, if required, are installed in **editable mode**. That is to say, any changes locally will take effect immediately.

1.1.5 Add dependencies to the subpackages

```
monas add click
```

The dependencies will be installed into the `.venv` folder under each subpackage.

1.1.6 Submit and push

```
git add .
git commit -m "initial commit"
git remote add origin <your repo url>
git push -u origin main
```

1.1.7 Bump version and Publish

```
monas bump
monas publish
```

A git tag of the specified version together with a PyPI release will be published.

1.2 Integration

Monas does not manage dependencies or environments itself, but it should work with most tools of Python packaging.

1.2.1 PEP 517 support

Since subpackages are installed in **editable mode**, Monas supports all build backends that support editable installation. The list includes:

1. setuptools
2. pdm
3. flit
4. hatch

Moreover, the above backends now all support PEP 621/PEP 631 metadata, so Monas will initialize the project metadata with PEP 621 format stored in `pyproject.toml`. At present, Monas does not work with package managers that do not support PEP 621.

1.2.2 Installer

Monas uses `virtualenv` and `pip` to install dependencies by creating a `.venv` folder under each subpackage. Fortunately, the above selected package managers can detect virtualenv automatically and you can start your work from it.

Note: Monas works the same in arbitrary sub directories in the project.

No lock files will be created

Monas, however, does not create any lock files. This is to ensure the installation is fast and correct. That is to say, Monas doesn't talk with any package managers other than `pip`.

Thanks to the standardization of PEP 621 and PEP 631, Monas is able to add new dependency lines into the `pyproject.toml` file in a uniform way. You can anyway use your favorite package managers from the subpackage directory and create lock files if necessary.

1.3 monas

1.3.1 monas

[cyan]Monas[/]: Python monorepo made easy

```
monas [OPTIONS] COMMAND [ARGS]...
```

Options

--version
: Show the version and exit.

add

Add a dependency(PEP 508 string) to specified packages.

```
monas add [OPTIONS] DEPENDENCY
```

Options

--no-install
: Do not automatically chain [primary]monas install[/] after added

-c, --concurrency <concurrency>
: The number of concurrent processes to use

--include <GLOB>
: [cyan](multiple)[/]Include packages matching the given glob

--exclude <GLOB>

[cyan](multiple)[/]Exclude packages matching the given glob

Arguments

DEPENDENCY

Required argument

bump

Bump version of all changed packages and release to git.

[bold]Example[/]: [green]0.1.0[/]

monas bump: [green]0.1.1[/]

monas bump major: [green]1.0.0[/]

monas bump minor: [green]0.2.0[/]

monas bump patch: [green]0.1.1[/]

monas bump alpha: [green]0.1.0alpha0[/]

monas bump dev: [green]0.1.0dev0[/]

monas bump post: [green]0.1.0post0[/]

monas major alpha: [green]1.0.0alpha0[/]

```
monas bump [OPTIONS] [PART] [TAG]
```

Options

--skip-git

Skip git operations

-m, --message <message>

Commit message

Arguments

PART

Optional argument

TAG

Optional argument

changed

List packages changed since last tagged release.

```
monas changed [OPTIONS]
```

Options

-l, --long
Show the full path

--json
Output in JSON format

init

Initialize the monas tool.

Args: path: The path to the *pyproject.toml* file.

```
monas init [OPTIONS]
```

Options

-p, --python <PYTHON_VERSION>
The Python version to use

-v, --version <version>
The version of the monorepo

install

Link the packages and install the remaining dependencies.

```
monas install [OPTIONS]
```

Options

-c, --concurrency <concurrency>
The number of concurrent processes to use

--root
Install all packages into the root project

--include <GLOB>
[cyan](multiple)[/]Include packages matching the given glob

--exclude <GLOB>
[cyan](multiple)[/]Exclude packages matching the given glob

list

List packages managed by Monas. [yellow]alias: ls[/]

```
monas list [OPTIONS]
```

Options

-l, --long
Show the full path

--json
Output in JSON format

--include <GLOB>
[cyan](multiple)[/]Include packages matching the given glob

--exclude <GLOB>
[cyan](multiple)[/]Exclude packages matching the given glob

new

Create a new <package> under <location>.

The package name must be locally unique and available on PyPI.

If location isn't given, it defaults to the first location of *packages* config.

```
monas new [OPTIONS] PACKAGE [LOCATION]
```

Arguments

PACKAGE
Required argument

LOCATION
Optional argument

publish

Publish packages in this release to PyPI.

```
monas publish [OPTIONS]
```

Options

-c, --concurrency <concurrency>
The number of concurrent processes to use

--no-sdist
Do not build sdist

-u, --username <username>
PyPI username

-p, --password <password>
PyPI password or token

-r, --repository <repository>
Repository name or URL

remove

Remove a dependency from specified packages.

```
monas remove [OPTIONS] DEPENDENCY
```

Options

--no-install
Do not automatically chain [primary]monas install[/] after added

-c, --concurrency <concurrency>
The number of concurrent processes to use

--include <GLOB>
[cyan](multiple)[/]Include packages matching the given glob

--exclude <GLOB>
[cyan](multiple)[/]Exclude packages matching the given glob

Arguments

DEPENDENCY
Required argument

1.4 Changelog

1.4.1 v0.0.3 (2022-04-17)

Features

- Add `remove` command.

Chores

- Add test suite.
- Add documentation.

1.5 Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

1.5.1 Environment setup

Nothing easier!

Fork and clone the repository:

```
git clone https://github.com/frostming/monas
cd monas
```

We use pdm to manage the project and dependencies, install PDM if it isn't done yet, then:

```
pdm install -d
```

You now have the dependencies installed.

You can run the tests with `pdm run test [ARGS...]`.

1.5.2 Test against multiple Python versions

This project uses nox as the test runner. See what sessions are list:

```
nox --list
```

And run the test suite on specified Python versions:

```
nox -s tests-3.8
```

!!! important "TIPS" `nox` and `pre-commit` in the following section are not list in the `dev-dependencies` of the project, because they can be installed separately to the system and used via the external executable. If you are willing to reproduce the development environment without external dependencies. Run `pdm add -d nox pre-commit` and the corresponding commands should be prefixed with `pdm run` as well.

1.5.3 Development

As usual:

1. create a new branch: `git checkout -b feature-or-bugfix-name`
2. edit the code and/or the documentation

If you updated the documentation or the project dependencies:

1. run `pdm run doc`

2. go to http://localhost:8000 and check that everything looks good

Before committing:

1. Make sure you submit a news entry under `news/` directory with the name pattern `<issue_or_pr_num>.<type>.md` where `<type>` should be one of:
    1. `bugfix` for bug fixes
    2. `feature` for features and improvements
    3. `doc` for documentation improvements
    4. `remove` for deprecations and removals
    5. `dep` for dependencies updates
    6. `misc` for miscellany tasks
2. Install pre-commit and hooks:

```
pre-commit install
```

3. Then linter task will be run each time when you commit something. Or you can run it manually:

```
pdm run lint
```

If you are unsure about how to fix or ignore a warning, just let the continuous integration fail, and we will help you during review.

Don't bother updating the changelog, we will take care of this.

1.5.4 Pull requests guidelines

Link to any related issue in the Pull Request message.

During review, we recommend using fixups:

```
# SHA is the SHA of the commit you want to fix
git commit --fixup=SHA
```

Once all the changes are approved, you can squash your commits:

```
git rebase -i --autosquash master
```

And force-push:

```
git push -f
```

If this seems all too complicated, you can push or force-push each new commit, and we will squash them ourselves if needed, before merging.

INDEX

Symbols

--concurrency
    monas-add command line option, 5
    monas-install command line option, 7
    monas-publish command line option, 9
    monas-remove command line option, 9
--exclude
    monas-add command line option, 5
    monas-install command line option, 7
    monas-list command line option, 8
    monas-remove command line option, 9
--include
    monas-add command line option, 5
    monas-install command line option, 7
    monas-list command line option, 8
    monas-remove command line option, 9
--json
    monas-changed command line option, 7
    monas-list command line option, 8
--long
    monas-changed command line option, 7
    monas-list command line option, 8
--message
    monas-bump command line option, 6
--no-install
    monas-add command line option, 5
    monas-remove command line option, 9
--no-sdist
    monas-publish command line option, 9
--password
    monas-publish command line option, 9
--python
    monas-init command line option, 7
--repository
    monas-publish command line option, 9
--root
    monas-install command line option, 7
--skip-git
    monas-bump command line option, 6
--username
    monas-publish command line option, 9
--version
    monas command line option, 5
    monas-init command line option, 7
-c
    monas-add command line option, 5
    monas-install command line option, 7
    monas-publish command line option, 9
    monas-remove command line option, 9
-l
    monas-changed command line option, 7
    monas-list command line option, 8
-m
    monas-bump command line option, 6
-p
    monas-init command line option, 7
    monas-publish command line option, 9
-r
    monas-publish command line option, 9
-u
    monas-publish command line option, 9
-v
    monas-init command line option, 7

D

DEPENDENCY
    monas-add command line option, 6
    monas-remove command line option, 9

L

LOCATION
    monas-new command line option, 8

M

monas command line option
    --version, 5
monas-add command line option
    --concurrency, 5
    --exclude, 5
    --include, 5
    --no-install, 5
    -c, 5
    DEPENDENCY, 6
monas-bump command line option

--message, 6
--skip-git, 6
-m, 6
PART, 6
TAG, 6
monas-changed command line option
--json, 7
--long, 7
-l, 7
monas-init command line option
--python, 7
--version, 7
-p, 7
-v, 7
monas-install command line option
--concurrency, 7
--exclude, 7
--include, 7
--root, 7
-c, 7
monas-list command line option
--exclude, 8
--include, 8
--json, 8
--long, 8
-l, 8
monas-new command line option
LOCATION, 8
PACKAGE, 8
monas-publish command line option
--concurrency, 9
--no-sdist, 9
--password, 9
--repository, 9
--username, 9
-c, 9
-p, 9
-r, 9
-u, 9
monas-remove command line option
--concurrency, 9
--exclude, 9
--include, 9
--no-install, 9
-c, 9
DEPENDENCY, 9

P

PACKAGE
monas-new command line option, 8
PART
monas-bump command line option, 6

T

TAG
monas-bump command line option, 6